BAB II

LANDASAN TEORI

1.1 Demam Berdarah Dengue

Demam berdarah (DB) atau demam berdarah dengue (DBD) adalah penyakit demam akut yang ditemukan di daerah tropis, dengan penyebaran geografis yang mirip dengan malaria. Penyakit ini disebabkan oleh salah satu dari empat serotipe virus dari *genus flavivirus*, *family flaviviridae*. Setiap serotipe cukup berbeda sehingga tidak ada proteksi-silang dan wabah yang disebabkan beberapa serotipe (*hiperendemisitas*) dapat terjadi. Demam berdarah disebarkan kepada manusia oleh nyamuk *aedes aegypti*.

Virus dengue penyakit ini ditunjukkan melalui munculnya demam tinggi terus menerus, disertai adanya tanda pendarahan, contohnya ruam. Ruam demam berdarah mempunyai ciri-ciri merah terang. Selain itu tanda dan gejala lainnya adalah sakit perut, rasa mual, *trombositopenia*, *hemokonsentrasi*, sakit kepala berat, sakit pada sendi (*artralgia*), sakit pada otot (*mialgia*). Sejumlah kecil kasus bisa menyebabkan sindrom *shock dengue* yang mempunyai tingkat kematian tinggi. Kondisi waspada ini perlu disikapi dengan pengetahuan yang luas oleh penderita maupun keluarga yang harus segera konsultasi ke dokter apabila pasien/penderita mengalami demam tinggi 3 hari berturut-turut. Banyak penderita atau keluarga penderita mengalami kondisi fatal karena menganggap ringan gejala-gejala tersebut.

Sesudah masa tunas/inkubasi selama 3-15 hari orang yang tertular dapat mengalami/menderita penyakit ini dalam salah satu dari 4 bentuk berikut ini: bentuk *abortif*, penderita tidak merasakan suatu gejala apapun.

Dengue klasik, penderita mengalami demam tinggi selama 4-7 hari, nyeri-nyeri pada tulang, diikuti dengan munculnya bintik-bintik atau bercak-bercak perdarahan dibawah kulit. *Dengue haemorrhagic fever* (demam berdarah dengue/DBD) gejalanya sama dengan dengue klasik ditambah dengan perdarahan dari hidung (*epistaksis*/mimisan), mulut, dubur, dsb. Dengue syok sindrom, gejalanya sama dengan DBD ditambah dengan syok/presyok. Bentuk ini sering berujung pada kematian.

Karena seringnya terjadi perdarahan dan syok maka pada penyakit ini angka kematiannya cukup tinggi, oleh karena itu setiap Penderita yang diduga menderita demam berdarah dalam tingkat yang manapun harus segera dibawa ke dokter atau puskesmas, mengingat sewaktu-waktu dapat mengalami syok/kematian. Demam berdarah umumnya lamanya sekitar enam atau tujuh hari dengan puncak demam yang lebih kecil terjadi pada akhir masa demam. Secara klinis, jumlah *platelet* akan jatuh hingga pasien dianggap *afebril*.

Diagnosis demam berdarah biasa dilakukan secara klinis. Biasanya yang terjadi adalah demam tanpa adanya sumber infeksi, ruam *petekial* dengan *trombositopenia* dan *leukopenia* relatif. *Serologi* dan reaksi berantai *polimerase* tersedia untuk memastikan diagnosa demam berdarah jika terindikasi secara

klinis. Mendiagnosis demam berdarah secara dini dapat mengurangi risiko kematian daripada menunggu akut.

Pencegahan untuk demam berdarah saat ini tidak ada vaksin yang tersedia secara komersial untuk penyakit demam berdarah. Pencegahan utama demam berdarah terletak pada menghapuskan atau mengurangi vektor nyamuk demam berdarah. Insiatif untuk menghapus kolam-kolam air yang tidak berguna (misalnya di pot bunga) telah terbukti berguna untuk mengontrol penyakit yang disebabkan nyamuk, menguras bak mandi setiap seminggu sekali, dan membuang hal-hal yang dapat mengakibatkan sarang nyamuk demam berdarah *aedes aegypti*.

Hal-hal yang harus dilakukan untuk menjaga kesehatan agar terhindar dari penyakit demam berdarah, sebagai berikut:

a. Melakukan kebiasaan baik, seperti makan makanan bergizi, rutin olahraga, dan istirahat yang cukup.
b. Memasuki masa pancaroba, perhatikan kebersihan lingkungan tempat tinggal dan melakukan 3M, yaitu: menguras bak mandi, menutup wadah yang dapat menampung air, dan mengubur barang-barang bekas yang dapat menjadi sarang perkembangan jentik-jentik nyamuk, meski pun dalam hal mengubur barang-barang bekas tidak baik, karena dapat menyebabkan polusi tanah. Akan lebih baik bila barang-barang bekas tersebut didaur-ulang.

c. *Fogging* atau pengasapan hanya akan mematikan nyamuk dewasa, sedangkan bubuk abate akan mematikan jentik pada air. Keduanya harus dilakukan untuk memutuskan rantai perkembangbiakan nyamuk.

d. Segera berikan obat penurun panas untuk demam apabila penderita mengalami demam atau panas tinggi.

e. Bagian terpenting dari pengobatannya adalah terapi *suportif*. Sang pasien disarankan untuk menjaga penyerapan makanan, terutama dalam bentuk cairan. Jika hal itu tidak dapat dilakukan, penambahan dengan cairan intravena mungkin diperlukan untuk mencegah dehidrasi dan hemokonsentrasi yang berlebihan. Transfusi *platelet* dilakukan jika jumlah *platelet* menurun drastis. Pengobatan alternatif yang umum dikenal adalah dengan meminum ekstrak daun jambu biji. Merujuk hasil kerja sama penelitian Fakultas Kedokteran Unair dan BPOM, ekstrak daun jambu biji bisa menghambat pertumbuhan virus dengue. Bahan itu juga meningkatkan trombosit tanpa efek samping. Masyarakat mesti memperhatikan informasi penting ini. Berdasarkan hasil kerja sama dalam uji pre klinis Fakultas Kedokteran Universitas Airlangga, Surabaya, Jawa Timur dan Badan Pengawas Obat dan Makanan (BPOM) yang dilansir di Jakarta, ekstrak daun jambu biji dipastikan bisa menghambat pertumbuhan virus dengue penyebab demam berdarah dengue (DBD). Bahan itu juga mampu meningkatkan jumlah trombosit hingga 100 ribu milimeter per kubik tanpa efek samping. Peningkatan tersebut diperkirakan dapat

tercapai dalam tempo delapan hingga 48 jam setelah ekstrak daun jambu biji dikonsumsi (http://id.wikipedia.org/wiki/Demam_berdarah).

1.2 Endemis

Pengertian endemis adalah suatu keadaan dimana suatu penyakit atau gen infeksi tertentu secara terus menerus ditemukan disuatu wilayah tertentu, bisa juga dikatakan sebagai suatu penyakit yang umum ditemukan di suatu wilayah. Pengertian *hyperendemis* adalah keadaan dimana penyakit tertentu selalu ditemukan di suatu wilayah dengan insiden yang tinggi. *Holoendemis* adalah keadaan dimana suatu penyakit selalu ditemukan di suatu wilayah dengan *prevalensi* yang tinggi, awalnya menyerang penduduk usia muda dan menimpa sebagian besar penduduk contohnya demam berdarah di daerah tertentu (Indrasanto, Doti. 2006).

1.3 *Incidence Rate*

Angka insidensi (*incidence rate*) adalah jumlah kasus baru penyakit tertentu yang dilaporkan pada periode waktu tertentu, tempat tertentu dibagi dengan jumlah penduduk dimana penyakit tersebut berjangkit. Biasanya dinyatakan dalam jumlah kasus per 1000 atau per 100.000 penduduk per tahun. Angka ini bisa diberlakukan bagi umur tertentu, jenis kelamin tertentu atau karakteristik spesifik dari penduduk (lihat angka morbiditas, angka prevalensi). *"attack rate"* atau *"case rate"* adalah proporsi yang menggambarkan insidensi kumulatif dari kelompok tertentu, yang diamati dalam waktu yang terbatas dalam

situasi tertentu misalnya pada waktu terjadi kejadian luar biasa atau wabah, dinyatakan dalam persentase (jumlah kasus per 100 penduduk).

Attack rate sekunder adalah jumlah penderita baru yang terjadi dalam keluarga atau institusi dalam periode masa inkubasi tertentu setelah terjadi kontak dengan kasus primer, dihubungkan dengan total keseluruhan kontak, deniominatornya/penyebutnya bisa terbatas hanya kepada kontak yang rentan saja, jika hal ini diketahui dengan jelas. Angka infeksi adalah proporsi yang menggambarkan insidensi dari semua infeksi yang terjadi baik yang kelihatan maupun yang tidak kelihatan (Indrasanto, Doti. 2006).

$$IR = \frac{\text{Jumlah kasus baru suatu penyakit selama periode tertentu}}{\text{Populasi yang mempunyairesiko}} \times 1000$$

Untuk menentukan tingkat kegawatan dapat ditentukan dari nilai *incidence rate*, dan membedakan nilai *incidence rate* tingkat kegawatan itu ditentukan bedasarkan warna. Untuk nilai dan warna dari tingkat kegawatan dapat dilihat pada Tabel 2.1 tabel tingkat kegawatan.

Tabel 2.1 Tabel Tingkat Kegawatan

No	Ratio Kenaikan Tingkat Kegawatan	Nilai IR	Tingkat Kegawatan
1.	8 x IR Target	> 337	Super Gawat
2.	6 x IR Target	> 223,5 - ≤ 337	Gawat
3.	4 x IR Target	> 55 - ≤ 223,5	Waspada
4.	1 x IR Target	≤ 55	Siaga

1.4 *Case Fataly Rate*

Case fataly rate (angka kematian kasus) adalah biasanya dinyatakan dalam persentase orang yang didiagnosa dengan penyakit tertentu kemudian meninggal karena penyakit tersebut dalam kurun waktu tertentu (Indrasanto, Doti. 2006).

$$CFR = \frac{\text{Jumlah kematian akibat penyakit dalam periode tertentu}}{\text{Jumlah Penyakit yang terdiaknosa dalam periode yang sama}} X100\%$$

1.5 Angka Bebas Jentik (ABJ)

Persentase rumah dan tempat umum yang tidak ditemukan jentik pada pemeriksaan jentik (Indrasanto, Doti. 2006).

$$ABJ = \frac{\text{Jumlah rumah tanpa jentik}}{\text{Jumlah rumah diperiksa}} X\ 100\ \%$$

Nilai ABJ ideal > 95 % .

1.6 Indikator *Attack Rate*

Indikator attack rate angka pengukuran yang dipakai untuk menghitung insiden kasus baru selama kejadian wabah. Angka serangan sekunder dihitung berdasarkan jumlah kasus baru yang sebelumnya mengadakan kontak dengan kasus primer (kasus yang mejadi sumber penularan) dalam masa inkubasi penyakit (Indrasanto, Doti. 2006).

$$\textit{Attack Rate} = \frac{\text{Jumlah penderita dbd dalam periode tertentu}}{\text{Jumlah penduduk terancam dbd dalam periode yang sama}} \textit{ x 100\%}$$

1.7 Sistem

Menurut Herlambang (2005:116), definisi sistem dapat dibagi menjadi dua pendekatan, yaitu pendekatan secara prosedur dan pendekatan secara komponen. Berdasarkan pendekatan prosedur, sistem didefinisikan sebagai kumpulan dari beberapa prosedur yang mempunyai tujuan tertentu. Sedangkan berdasarkan pendekatan komponen, sistem merupakan kumpulan dari komponen–komponen yang saling berkaitan untuk mencapai tujuan tertentu.

Dalam perkembangan sistem yang ada, sistem yang dibedakan menjadi dua jenis, yaitu sistem terbuka dan sistem tertutup. Sistem terbuka merupakan sistem yang dihubungkan dengan arus sumber daya luas dan tidak mempunyai elemen pengendali. Sedangkan sistem tertutup tidak mempunyai elemen pengontrol dan dihubungkan pada lingkungan sekitarnya.

1.8 Aplikasi *Web*

Aplikasi *web* atau sering disebut *web application* merupakan aplikasi yang dibuat dengan menggunakan bahasa pemograman *web*. Kebanyakan bahasa pengrograman *web* berbasis *server*, namun tidak menutup kemungkinan digunakan bahasa pemrograman *web* berbasis *client*. *Web server* atau web *application* berbeda dengan situs *web* biasa (*homepage*), *web application* lebih dinamis dan atraktif serta dapat mengelola data dengan baik.

Bahasa pemograman untuk aplikasi *web* yang ada cukup beragam tapi beberapa diantaranya yang sering digunakan adalah PHP, JSP, PERL/CGI, ASP

dan *coldfusion*. Bahasa pemrograman *web* yang berbasis *client* diantaranya adalah *java, java script, VB script.*

Aplikasi *web* maksudnya adalah aplikasi web yang hanya diproses di *server* atas permintaan *client* dan mengirim hasil proses ke *client*. *Source code* yang dibuat tidak akan tampak pada *client*, yang tampak hanyalah script-script *client* seperti *java, java script,* VB *script*. Aplikasi *web* dapat berjalan dengan bantuan *web server* atau *web application server*.

Web server adalah sebuah program yang dijalankan pada komputer *server*, yang bertugas menyediakan jasa pelayanan internet kepada komputer-komputer yang terhubung ke *server*. *Web server* menggunakan protokol TCP/IP yang bersifat terbuka sehingga dapat menggabungkan kombinasi perangkat keras, perangkat lunak, dan sistem operasi yang digunakan. *Web server* tidak banyak melakukan tugas pemrosesan, melainkan hanya melayani permintaan dari komputer *client*. *Server* juga berfungsi menyimpan program dan *file-file* yang dibutuhkan oleh semua komputer yang terhubung kepadanya. Jadi setiap kali ada permintaan dari komputer *client*, program *web server* akan mencari *file* yang diminta pada *hard disk server*, lalu memberikannya kepada *client* yang memintanya.

2.9 *Data Flow Diagram*

Data Flow Diagram (DFD) adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data

tersebut. Kita dapat menggunakan DFD untuk dua hal utama, yaitu untuk membuat dokumentasi dari sistem informasi yang ada, atau untuk menyusun dokumentasi untuk sistem informasi yang baru.

2.9.1 *Context Diagram* (CD)

Jenis pertama *context diagram*, adalah data *flow* diagram tingkat atas (DFD *top level*), yaitu diagram yang paling tidak detail, dari sebuah sistem informasi yang menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar entitas-entitas eksternal. (CD menggambarkan sistem dalam satu lingkaran dan hubungan dengan entitas luar. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem).

2.9.2 DFD Fisik

Adalah representasi grafik dari sebuah sistem yang menunjukan entitas-entitas internal dan eksternal dari sistem tersebut, dan aliran-aliran data ke dalam dan ke luar dari entitas-entitas tersebut. Entitas-entitas internal adalah personel, tempat (sebuah bagian), atau mesin (misalnya, sebuah komputer) dalam sistem tersebut yang mentransformasikan data. Maka DFD fisik tidak menunjukkan apa yang dilakukan, tetapi menunjukkan dimana, bagaimana, dan oleh siapa proses-proses dalam sebuah sistem dilakukan.

Perlu diperhatikan didalam memberikan keterangan di lingkaran-lingkaran (simbol proses) dan aliran-aliran data (simbol aliran data) dalam DFD fisik

menggunakan label/keterangan dari kata benda untuk menunjukan bagaimana sistem mentransmisikan data antara lingkaran-lingkaran tersebut.

2.9.3 DFD *Logis*

Adalah representasi grafik dari sebuah sistem yang menunjukkan proses-proses dalam sistem tersebut dan aliran-aliran data ke dalam dan ke luar dari proses-proses tersebut. Kita menggunakan DFD *logis* untuk membuat dokumentasi sebuah sistem informasi karena DFD *logis* dapat mewakili logika tersebut, yaitu apa yang dilakukan oleh sistem tersebut, tanpa perlu menspesifikasi dimana, bagaimana, dan oleh siapa proses-proses dalam sistem tersebut dilakukan. Keuntungan dari DFD logis dibandingkan dengan DFD fisik adalah dapat memusatkan perhatian pada fungsi-funsi yang dilakukan sistem (Sidarta, Lani. 1995).

2.10 *Entity Relationship Diagram* (ERD)

Entity relationship diagram merupakan jaringan yang menggunakan susunan data yang disimpann dari sistem secara abstrak. Tujuan dari *entity relationship* adalah untuk menunjukkan objek data dan *relationship* yang ada pada objek tersebut (Sidarta, Lani. 1995).

Komponen (Simbol) ERD:

1. *Entity*

   Adalah suatu objek yang dapat dibedakan atau dapat diidentifikasikan secara unik dengan objek lainnya, dimana semua informasi yang berkaitan dengannya dikumpulkan.

2. *Relationship*

   Adalah hubungan yang terjadi antara satu *entity* dengan *entity* lainnya. *Relationship* tidak mempunyai keberadaan fisik atau konseptual kecuali yang sejenis dinamakan dengan *relationship* diagram.

3. Atribut

   Adalah karakteristik dari *entity* atau *relationship* yang menyediakan penjelasan detail tentang *entity* atau *relationship* tersebut.

2.10.1 Derajat *Relationship* adalah:

1. *Unary* (derajat satu) adalah satu buah *relationship* menghubungkan satu buah *entity*.
2. *Binary* (derajat dua) adalah satu buah *relationship* yang menghubungkan dua buah *entity*.
3. *Ternary* (derajat tiga) adalah satu buah *relationship* menghubungkan tiga buah *entity*.

2.10.2 *Cardinality* Rasio

Yaitu menjelaskan batasan pada jumlah *entity* yang berhubungan melalui suatu *relationship*.

Jenis-jenis *Cardinality* Rasio:

1. *One To One* (1:1)

   Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding satu berbanding satu.

2. *One To Many* (1:M)

   Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding satu berbanding banyak.

3. *Many To One* (M:1)

   Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding banyak berbanding satu.

4. *Many To Many* (M:M)

   Yaitu perbandingan antara *entity* pertama dengan *entity* kedua berbanding banyak berbanding banyak.

2.11 *White Box Testing*

White box testing adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara procedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Secara sekilas dapat diambil kesimpulan *white box testing* merupakan petunjuk untuk mendapatkan program yang benar secara 100%. Pengujian dilakukan berdasarkan bagaimana suatu *software* menghasilkan *output* dari *input*. Pengujian ini dilakukan berdasarkan kode program (Romeo, 2006).

2.12 *Black Box Testing*

Black box testing adalah pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak. Jadi dianalogikan seperti kita melihat suatu kotak hitam, kita hanya bisa melihat penampilan luarnya saja, tanpa tau ada apa dibalik bungkus hitamnya. Sama seperti pengujian *black box*, mengevaluasi hanya dari tampilan luarnya (*interface* nya) , fungsionalitasnya. Tanpa mengetahui apa sesungguhnya yang terjadi dalam proses detilnya (hanya mengetahui *input* dan *output*).

Black Box testing adalah metode pengujian perangkat lunak yang menguji fungsionalitas aplikasi yang bertentangan dengan struktur internal atau kerja. Pengetahuan khusus dari kode aplikasi/struktur internal dan pengetahuan pemrograman pada umumnya tidak diperlukan. Uji kasus dibangun di sekitar spesifikasi dan persyaratan, yakni, aplikasi apa yang seharusnya dilakukan. Menggunakan deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus. Tes ini dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Perancang uji memilih input yang *valid* dan tidak *valid* dan menentukan *output* yang benar. Tidak ada pengetahuan tentang struktur internal benda uji itu.

Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan. Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit *testing* juga (Romeo, 2006)

Pengujian pada *Black Box* berusaha menemukan kesalahan seperti:

a. Fungsi-fungsi yang tidak benar atau hilang.
b. Kesalahan *interface*.
c. Kesalahan dalam struktur data atau akses database eksternal.
d. Kesalahan kinerja.
e. Inisialisasi dan kesalahan terminasi.

2.13 Puskesmas

Puskesmas adalah unit pelaksana teknis dinas kesehatan/kabupaten kota yang bertanggung jawab menyelenggarakan pembangunan kesehatan di suatu wilayah kerja. Upaya puskesmas diharapkan dapat mewujudkan kecamatan sehat menuju Indonesia sehat dan bertanggung jawab menyelenggarakan upaya kesehatan perorangan dan upaya kesehatan masyarakat yang dikelompokkan menjadi upaya kesehatan wajib dan upaya kesehatan pengembangan (Harbakti Rasa, 2013). Pengertian puskesmas antara lain:

a. Unit pelaksana pembangunan kesehatan di wilayah kecamatan.
b. Satu satuan organisasi yang diberikan kemandirian oleh dinas kesehatan. kabupaten/kodya untuk melaksanakan tugas-tugas operasional pembangunan kesehatan di wilayah kecamatan.
c. Pembangunan kesehatan.
d. Penyelenggaraan upaya kesehatan (kesadaran, kemauan, kemampuan sehat optimal.

2.14 Prototipe

Prototipe merupakan salah satu metode pengembangan perangkat lunak yang banyak digunakan. Dengan metode prototipe 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 dibutuhakan 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 menyelasaikan sistem yang diinginkan. Dengan demikian akan menghasilkan sistem sesuai dengan jadwal waktu penyelesaian yang telah ditentukan.

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

Tahapan-tahapan prototipe:

a. Pengumpulan kebutuhan.

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

b. Membangun prototipe.

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

c. Evaluasi protoptipe.

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

d. Mengkodekan sistem.

Dalam tahap ini prototipe yang sudah disepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.

e. Menguji sistem.

Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus dites dahulu sebelum digunakan. Pengujian ini dilakukan dengan *white box*, *black box*, *basis path*, pengujian arsitektur dan lain-lain.

f. Evaluasi sistem.

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

g. Menggunakan sistem.

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

Keunggulan dan kelemahan dari prototype:

a. Keunggulan prototipe adalah:
   1. Adanya komunikasi yang baik antara pengembang dan pelanggan.
   2. Pengembang dapat bekerja lebih baik dalam menentukan kebutuhan pelanggan.
   3. Pelanggan berperan aktif dalam pengembangan sistem.
   4. Lebih menghemat waktu dalam pengembangan sistem.
   5. Penerapan menjadi lebih mudah karena pemakai mengetahui apa yang diharapkannya.

b. Kelemahan prototipe adalah :
   1. Pelanggan kadang tidak melihat atau menyadari bahwa perangkat lunak yang ada belum mencantumkan kualitas perangkat lunak secara keseluruhan dan juga belum memikirkan kemampuan pemeliharaan untuk jangka waktu lama.

2. Pengembang biasanya ingin cepat menyelesaikan proyek. Sehingga menggunakan algoritma dan bahasa pemrograman yang sederhana untuk membuat prototipe lebih cepat selesai tanpa memikirkan lebih lanjut bahwa program tersebut hanya merupakan cetak biru sistem .
3. Hubungan pelanggan dengan komputer yang disediakan mungkin tidak mencerminkan teknik perancangan yang baik.

Prototipe bekerja dengan baik pada penerapan-penerapan yang berciri sebagai berikut:

1. Resiko tinggi yaitu untuk masalah-masalah yang tidak terstruktur dengan baik, ada perubahan yang besar dari waktu ke waktu, dan adanya persyaratan data yang tidak menentu.
2. Interaksi pemakai penting. Sistem harus menyediakan dialog *online* antara pelanggan dan komputer.
3. Perlunya penyelesaian yang cepat.
4. Perilaku pemakai yang sulit ditebak.
5. Sistem yang inovatif. Sistem tersebut membutuhkan cara penyelesaian masalah dan penggunaan perangkat keras yang mutakhir (Edward Aditya, 2010).