

BAB II

LANDASAN TEORI

2.1 Rekayasa Perangkat Lunak

2.1.1 Definisi Perangkat Lunak

Perangkat Lunak adalah program komputer yang menyediakan fitur-fitur, fungsi, dan kinerja yang dibutuhkan dalam memproses informasi yang dibutuhkan oleh pengguna ketika dijalankan (Pressman, Roger S, 2010).

Perangkat Lunak adalah instruksi yang langsung dijalankan komputer untuk melakukan pekerjaan dan terdapat pada berbagai bentuk aplikasi (Simarmata, Janner, 2010).

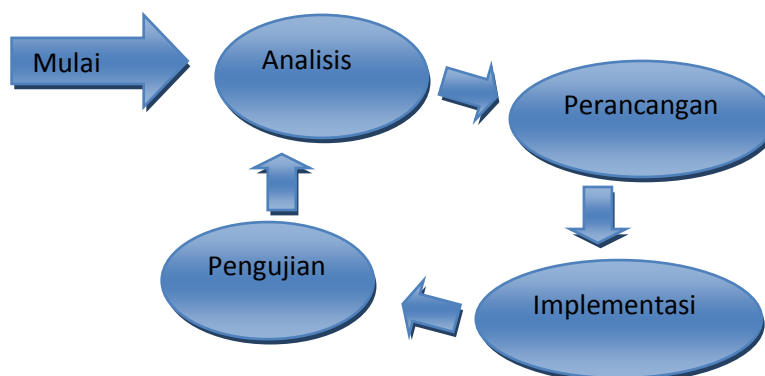
2.1.2 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak (RPL atau SE [*Software Engineering*]) adalah serangkaian aktivitas untuk menghasilkan perangkat lunak yang diinginkan dimulai dari pembuatan, pemeliharaan, manajemen organisasi pengembangan perangkat lunak dan sebagainya (S, Rosa A. dan M. Shalaluddin, 2013 : 4).

“Rekayasa perangkat lunak (*software engineering*) adalah pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomi yang dipercaya dan bekerja secara efisien menggunakan mesin” (Simarmata, Janner, 2010).

2.1.3 Proses Rekayasa Perangkat Lunak

Proses-proses yang dilakukan selama pembangunan perangkat lunak secara garis besar adalah sebagai berikut (S, Rosa A dan M. Shalahuddin, 2013 :11):



Sumber : S, Rosa. A dan M. Shalahuddin, 2013:11

Gambar 2.1 Tahapan Umum Rekayasa Perangkat Lunak

Proses perangkat lunak (*software process*) adalah sekumpulan aktifitas yang terus berulang dengan tujuan untuk mengembangkan atau mengubah perangkat lunak sehingga terjadi peningkatan kemampuan dan memenuhi kebutuhan pemakai. Secara umum proses perangkat lunak terdiri dari (S, Rosa A dan M. Shalahuddin, 2013:11) :

1. Pengumpulan Spesifikasi (*Spesification*), Mengetahui apa saja yang harus dikerjakan sistem perangkat lunak dan batasan pengembangan perangkat lunak.
2. Pengembangan (*Development*), Pengembangan perangkat lunak untuk menghasilkan sistem perangkat lunak.
3. Validasi (*Validation*), Memeriksa apakah perangkat lunak sudah memenuhi keinginan pelanggan (*costumer*).
4. Evolusi (*Evolution*), Mengubah perangkat lunak untuk memenuhi perubahan kebutuhan pelanggan (*customer*)

2.2 SDLC

2.2.1 Pengertian SDLC

“SDLC atau *Software Development Life Cycle* atau sering disebut juga *Sistem Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan *best practice* atau cara-cara yang sudah teruji baik)” (S, Rosa A. dan M. Shalahuddin, 2013 : 25).

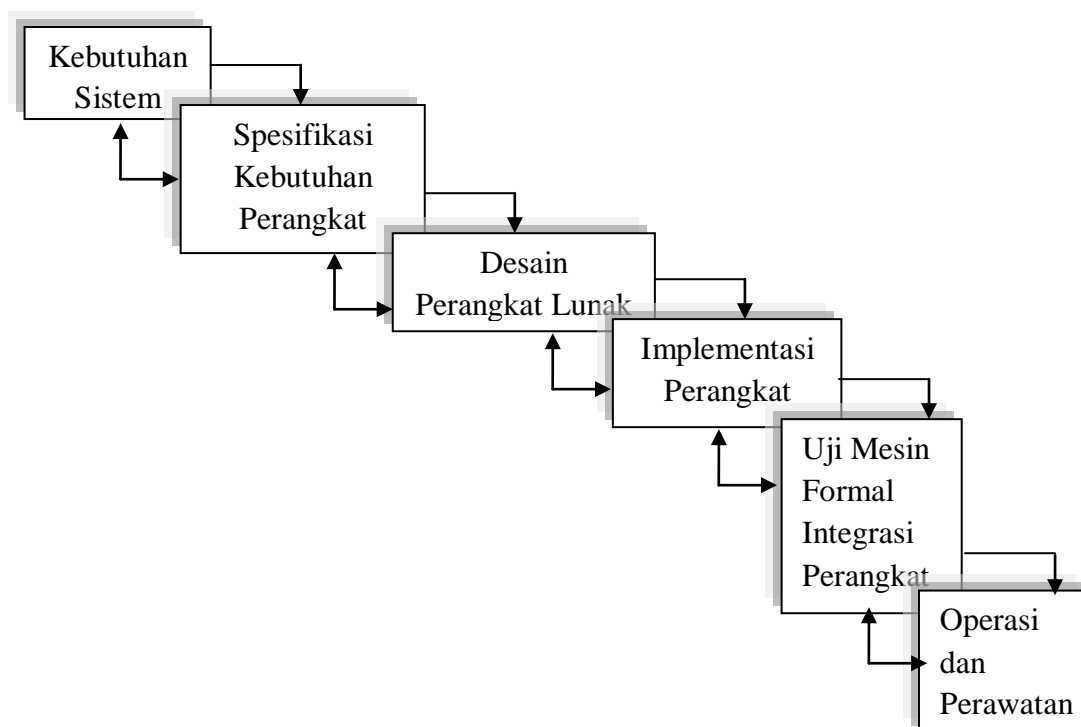
2.2.2 Model-Model Proses Pengembangan Perangkat Lunak

Secara umum model-model proses pengembangan perangkat lunak akan dimulai dari model air terjun (*waterfall model*) dan dilanjutkan dengan pendekatan prototipe, model spiral, proses pengembangan iteratif, dan pendekatan proses beorientasi objek (Simarmata, Janner, 2010 : 53).

Berikut model-model proses pengembangan perangkat lunak yang akan disajikan dalam kerangka kematangan proses, termasuk *capability Maturity Model* milik *software Engineering Institute* dan pendekatan dalam *software Productivity Research* serta dua bentuk standar kualitas, yaitu *The Malcolm Baldrige Assesment Dicipline* dan ISO 9000 (Simarmata, Janner, 2010 : 53) :

1. Model Pengembangan Air Terjun

Boehm (1976), model air terjun muncul dengan alasan untuk membantu mengatasi kerumitan yang terjadi akibat proyek-proyek pengembangan perangkat lunak. Seperti yang terlihat pada gambar 2.2, sebuah model air terjun memacu tim pengembang untuk merinci apa yang seharusnya perangkat lunak lakukan (mengumpulkan dan menentukan kebutuhan sistem) sebelum sistem tersebut dikembangkan .



Sumber :Simarmata, Janner, 2010: 54

Gambar 2.2 Penyajian Sederhana dari model pengembangan air terjun

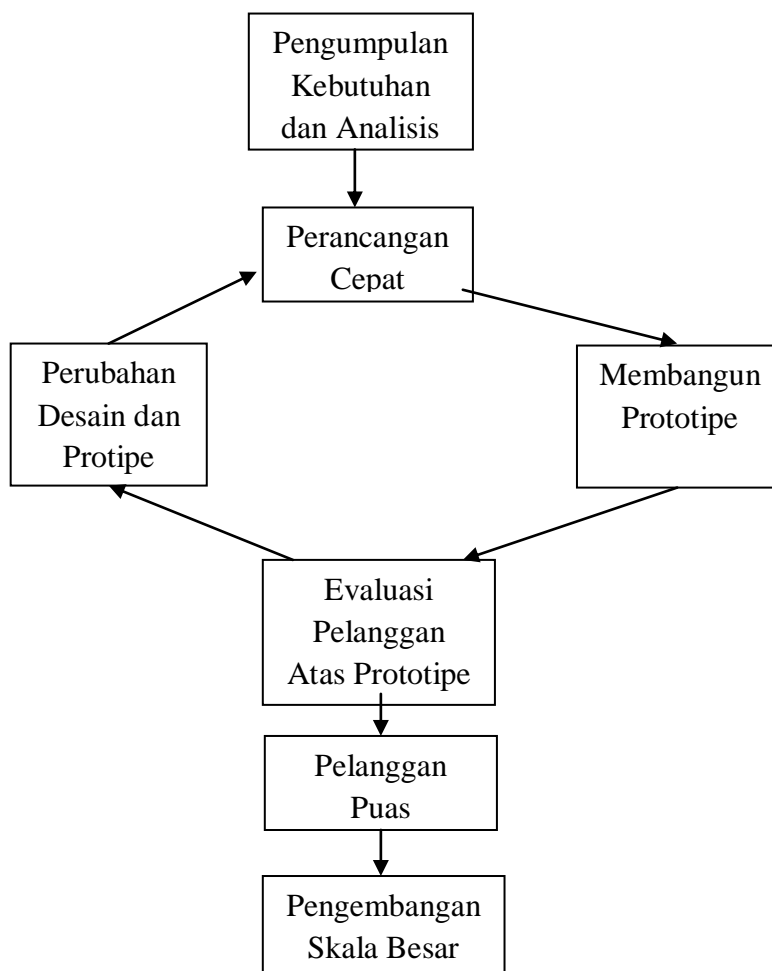
Kemudian, model ini memungkinkan pemecahan misi pengembangan yang rumit menjadi beberapa langkah logis (desain, kode, pengujian, dan seterusnya) dengan beberapa langkah yang pada akhirnya akan menjadi produk akhir yang siap pakai.

Tahapan disebut analisis kebutuhan data konsumen, pengguna; tahap pengembangan yang juga dapat dibagi menjadi beberapa desain tingkat tinggi dan desain tingkat terperinci; tahapan implementasi yang biasa disebut *code* dan *debug*; dan tahapan pengujian yang juga mencakup uji tingkat komponen, uji tingkat produk, dan uji tingkat sistem.

2. Pendekatan Prototipe

Sebuah prototipe adalah bagian dari produk yang mengekspresikan logika maupun fisik antarmuka *eksternal* yang ditampilkan. Gambar 2.3 menunjukkan pendekatan prototipe pada umumnya dan melibatkan beberapa langkah berikut:

1. Mengumpulkan dan menganalisis kebutuhan,
2. Melakukan perancangan cepat,
3. Membangun sebuah prototipe,
4. Evaluasi dilakukan oleh konsumen atas prototipe,
5. Perubahan rancangan dan prototipe,
6. Apabila pelanggan kecewa dengan prototipe yang telah dibangun, ulangi langkah 5, dan
7. Apabila pelanggan puas dengan prototipe yang telah dibangun, pengembangan produk skala besar dapat dimulai.



Sumber : *Simarmata, Janner, 2010 :62*

Gambar 2.3 Pendekatan Prototipe

Faktor kritis untuk kesuksesan pendekatan prototipe adalah perubahan cepat di dalam rancangan dan pembangunan prototipe. Pendekatan prototipe sangat sesuai untuk proyek kecil atau pada subsistem.

1) *Rapid Throwaway Prototype*

Pendekatan pengembangan perangkat keras/lunak ini dipopulerkan oleh Gomaa dan Scoot (1981) yang saat ini telah digunakan secara oleh industri, terutama di dalam pengembangan aplikasi. Pendekatan ini biasanya digunakan dengan item yang beresiko tinggi (*high-risk*) atau dengan bagian dari sistem yang tidak dimengerti secara keseluruhan oleh para tim pengembang. Pada pendekatan ini, prototipe “*quick and dirty*” dibangun, diverifikasi oleh konsumen, dan dibuang hingga prototipe yang diinginkan tercapai pada saat proyek berskala besar dimulai.

2) *Prototipe Evolusioner*

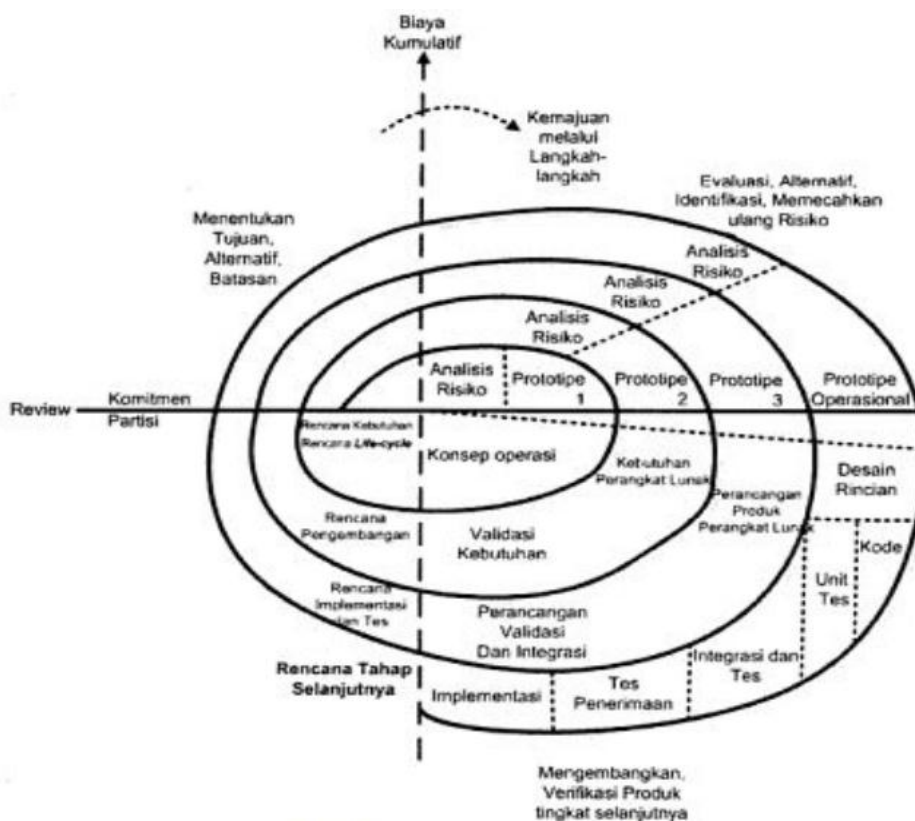
Hough (1993), pendekatan *evolusioner* adalah suatu prototipe yang dibangun berdasarkan pada kebutuhan dan pemahaman secara umum. Prototipe kemudian diubah daripada dibuang. Prototipe yang dibuang biasanya digunakan dengan aspek sistem yang dimengerti secara luas dan dibangun atas kekuatan tim pengembang. Prototipe ini juga didasarkan atas kebutuhan prioritas, kadang-kadang diacu sebagai “*chunking*” pada pengembangan aplikasi.

3. Model Spiral

Model ini menekankan pada pembuatan prototipe dan manajemen risiko yang sangat fleksibel jika dibandingkan dengan model air terjun. Dasar-dasar konsep model ini adalah bahwa setiap bagian produk dan setiap tingkatan melibatkan urutan yang sama pada setiap langkah(siklus). Dimulai dari tengah spiral, dapat dilihat bahwa setiap tahapan pengembangan (konsep operasi, kebutuhan perangkat lunak,

perancangan produk, rancangan detail, dan implementasi) melibatkan satu putaran (siklus) dari spiral.

Langkah awal dari setiap putaran dari spiral adalah untuk identifikasi tujuan dari produk yang diteliti. Disini, alternatif berarti implementasi dari bagian tersebut dan batasan mendesak pada aplikasi alternatif. Langkah selanjutnya adalah mengevaluasi alternatif untuk tujuan batasan, untuk mengidentifikasi hubungan yang beresiko dan menyelesaikannya. Analisis risiko dan pendekatan yang dikendalikan risiko menjadi karakteristik utama dari model spiral daripada pendekatan yang dikendalikan dokumen dari model air terjun. Gambar 2.4 menunjukkan model spiral Boehm .



Sumber : Simarmata, Janner, 2010 : 65

Gambar 2.4 Model Spiral dalam proses perangkat lunak

4. Model Proses Pengembangan Iteratif

Pendekatan *Iterative Enhancement* (IE) (Basili dan Turner, 1975) atau *Iterative development Process* (IDP) telah ditetapkan untuk dimulai dengan subset kebutuhan dan pengembangan sebuah subset dari produk yang memuaskan kebutuhan utama pelanggan, memberikan pengalaman untuk berkembang.

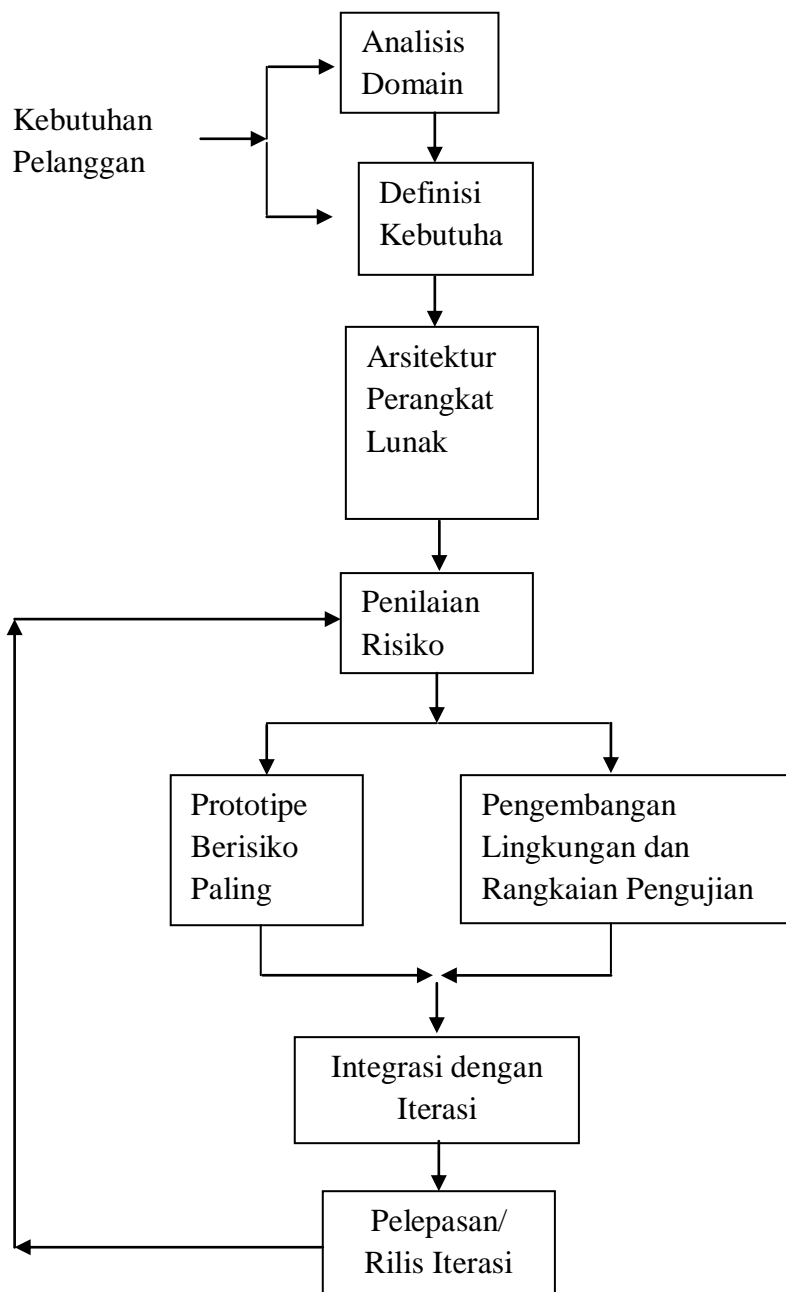
Model IDP merupakan penggabungan pembuatan prototipe (*prototyping*) dengan model air terjun klasik (*classical waterfall model*). Metode Analisis domain dan analisis risiko dapat pula dimasukkan ke dalam model IDP yang dapat disamakan dengan model spiral terutama pada pembuatan prototipe dan manajemen risiko.

Model IDP menggabungkan pembuatan prototipe (*prototyping*) dengan kekuatan dari model air terjun klasik (*classical waterfall model*). Metode-metode lainnya seperti analisis domain dan analisis risiko, juga dapat dimasukkan ke dalam model IDP. Model tersebut dapat disamakan dengan model spiral, terutama pada pembuatan prototipe dan manajemen risiko. Tentu saja, model spiral dianggap sebagai sebuah model IDP yang khusus, walaupun istilah IDP masih menjadi arti umum yang merupakan jenis-jenis bentuk dari model yang dapat dibuat.

Gambar 2.5 menunjukkan sebuah contoh dari model proses pengembangan iteratif yang digunakan oleh IBM Owego, New York, dengan tujuan untuk “membangun sebuah sistem dengan pengembangan sebuah prototipe arsitektur melalui serangkain versi yang mampu dieksekusi, dengan masing-masing pengalaman penggabungan iteratif yang sukses dan dengan lebih banyak penggunaan sistem”.

Luckey (2012), beberapa contoh implementasi yang terdiri dari delapan langkah utama :

1. Analisis domain,
2. Definisi kebutuhan,
3. Arsitektur perangkat lunak,
4. Penilaian risiko,
5. Prototipe,
6. Pengembangan lingkungan dan rangkaian pengujian (*test suite*),
7. Integrasi dengan iterasi yang sebelumnya, dan
8. Pelepasan iterasi.



Sumber : Simarmata, Janner, 2010 : 70

Gambar 2. 5 Contoh Model Proses Pengembangan Iteratif

Seperti yang diilustrasikan pada gambar 2.5, proses iterasi melibatkan lima langkah terakhir. Analisis domain, definisi kebutuhan, dan arsitektur perangkat lunak adalah langkah-langkah pra-iterasi yang serupa dengan langkah-langkah yang ada pada

model air terjun. Selama lima langkah iterasi tersebut dilakukan, aktivitas-aktivitas berikut akan berlangsung:

1. Menganalisis atau meninjau kebutuhan-kebutuhan sistem,
2. Merancang atau merevisi solusi yang terbaik untuk memenuhi kebutuhan-kebutuhan tersebut,
3. Mengenali risiko-risiko tertinggi bagi proyek dan mendahulukannya. Mengurangi risiko prioritas yang paling tinggi melalui prototipe, meninggalkan risiko yang lebih rendah untuk bagian iterasi berikutnya,
4. Menentukan dan menjadwalkan, atau meninjau ulang iterasi selanjutnya,
5. Mengembangkan lingkungan untuk pengujian iterasi yang sesuai dan mendukung lingkungan tes,
6. Menerapkan bagian perancangan yang sedikit dibutuhkan untuk memenuhi iterasi yang sedang berlangsung,
7. Menyatukan perangkat lunak dalam lingkungan pengujian dan melakukan pengujian secara regresi (mundur),
8. Membarui dokumen untuk diluncurkan bersamaan dengan iterasi, dan
9. Melepaskan/meluncurkan iterasi.

Sebagai catatan, pengembangan rangkaian pengujian (*test suite*) sepanjang perancangan dan pengembangan sangat penting untuk verifikasi (pembuktian) terhadap fungsi dan kualitas masing-masing iterasi. Meskipun demikian, aktivitas ini tidak selalu ditekankan secara jelas dalam praktiknya.

5. Proses Pengembangan Berorientasi Objek

Pendekatan berorientasi objek (*Objek-Oriented* [OO]) untuk perancangan dan pemrograman diperkenalkan pada tahun 1980an. Pendekatan ini mewakili suatu pergeseran paradigma utama dalam pengembangan perangkat lunak. Pendekatan ini selanjutnya akan menimbulkan akibat utama dalam perangkat lunak dalam beberapa tahun selanjutnya. Berbeda dengan pemrograman tradisional yang memisahkan data dan objek, pemrograman berorientasi objek dilakukan berdasarkan objek, dengan sekumpulan data yang telah ditetapkan dan sekumpulan operasi-operasi (metode) yang dapat dilaksanakan dalam data tersebut. Seperti paradigma dari perancangan struktural dan dekomposisi fungsional, pendekatan berorientasi objek telah menjadi sebuah dasar utama dari rekayasa perangkat lunak (*software engineering*) .

Branson dan Herness (1992), mengusulkan suatu proses pengembangan OO untuk proyek skala besar yang berpusat pada metodologi yang mempunyai delapan langkah dan didukung oleh sebuah mekanisme untuk penjajakan (*tracking*), serangkaian inspeksi (penyelidikan), sekumpulan teknologi, dan aturan-aturan untuk pembuatan prototipe dan pengujian. Proses pengembangan yang terdiri dari delapan langkah tersebut dibagi ke dalam tiga tahap logika, yaitu :

1. Tahap analisis

Tahap ini melakukan pengumpulan dan penyajian kebutuhan-kebutuhan pelanggan dengan cara yang singkat, untuk menggambarkan sebuah sistem yang mewakili kebutuhan-kebutuhan pengguna yang kurang diperhatikan dengan *platform* implementasi (lingkungan perangkat keras maupun perangkat lunak) yang telah dikembangkan.

2. Tahap perancangan

Tahap ini melakukan perubahan sistem penting sehingga sistem tersebut dapat diterapkan pada sekumpulan perangkat keras dan perangkat lunak yang diberikan. Kelas-kelas penting dan kelas-kelas inkarnasi digabung dan diperhalus ke dalam hirarki kelas pengembangan. Tujuan dari sintesis kelas adalah untuk mengoptimalkan penggunaan ulang (*reuse*) dan menciptakan kelas-kelas yang dapat digunakan kembali (*reusable*).

3. Tahap implementasi

Tahap ini mengambil kelas-kelas yang ditetapkan untuk penyelesaian.

Kedelapan langkah dari proses tersebut adalah sebagai berikut:

1) Model sistem penting

Sistem penting menggambarkan aspek-aspek yang dibutuhkan sistem untuk mencapai tujuannya, memperhatikan lingkungan perangkat keras dan perangkat lunak target. Model ini terdiri dari aktivitas penting dan data penting.

Langkah ini terdiri dari lima sublangkah, yaitu

- a. Membuat pandangan pengguna (*user view*),
- b. Membuat model aktivitas penting
- c. Menetapkan data solusi,
- d. Memperhalus model penting, dan
- e. Membangun analisis yang terperinci.

Langkah-langkah tersebut berpusat pada kebutuhan-kebutuhan pengguna, yaitu analisis, pembelahan, pembersihan, penggabungan, dan pengorganisasian ke dalam sebuah model logika penting dari sistem tersebut. Model ini berdasarkan pada pendapatn teknologi yang sempurna.

2) Memperoleh kelas calon penting

Langkah ini menggunakan sebuah teknik yang dikenal sebagai teknik “*carving*” (mengukir) untuk mengidentifikasi metode-metode dan calon-calon kelas penting dari model penting seluruh sistem. Sekumpulan diagram aliran data (*data-flow diagram*) yang lengkap, bersama dengan dukungan spesifikasi proses dan masukan kamus data adalah dasar untuk kelas dan pemilihan metode. Metode-metode dan calon-calon kelas ditemukan pada entitas eksternal, penyimpanan data, aliran masukan, dan spesifikasi proses.

3) Membatasi model penting

Model penting dimodifikasi untuk bekerja dalam batasan lingkungan implementasi target. Aktivitas penting dan data penting dialokasikan ke berbagai jenis proses dan container (*repository data*). Aktivitas-aktivitas kemudian ditambahkan ke sistem seperlunya, berdasarkan pada pembatasan di dalam lingkungan implementasi target yang di acu sebagai model inkarnasi.

4) Memperoleh kelas-kelas tambahan

Metode-metode dan kelas-kelas calon tambahan dikhususkan untuk lingkungan implementasi yang diseleksi berdasarkan aktivitas-aktivitas yang ditambahkan saat membatasi model penting. Kelas-kelas ini menyediakan antarmuka-antarmuka untuk kelas-kelas penting pada suatu level konsisten.

5) Menyatukan kelas-kelas

Calon-calon kelas penting dan calon-calon kelas tambahan dibersihkan dan diorganisasikan ke dalam suatu *hierarki*. Atribut-atribut dan operasi-operasi umum digali untuk menghasilkan beberapa superkelas dan subkelas. Kelas-kelas akhir dipilih untuk memaksimalkan penggunaan ulang melalui pewarisan dan pemasukan.

6) Menetapkan antarmuka-antarmuka

Antarmuka-antarmuka (interface), deklarasi tipe objek, dan definisi kelas ditulis berdasarkan penyatuan kelas-kelas yang didokumentasikan.

7) Menyelesaikan rancangan

Pada langkah ini, rancangan modul (*module*) implementasi akan diselesaikan. Modul implementasi meliputi beberapa metode yang masing-masing memberikan satu fungsi kohesif tunggal. Logika, interaksi sistem, dan harapan-harapan metode pada kelas-kelas lain digunakan untuk memenuhi rancangan lengkap masing-masing metode dalam sebuah kelas.

8) Menerapkan solusi

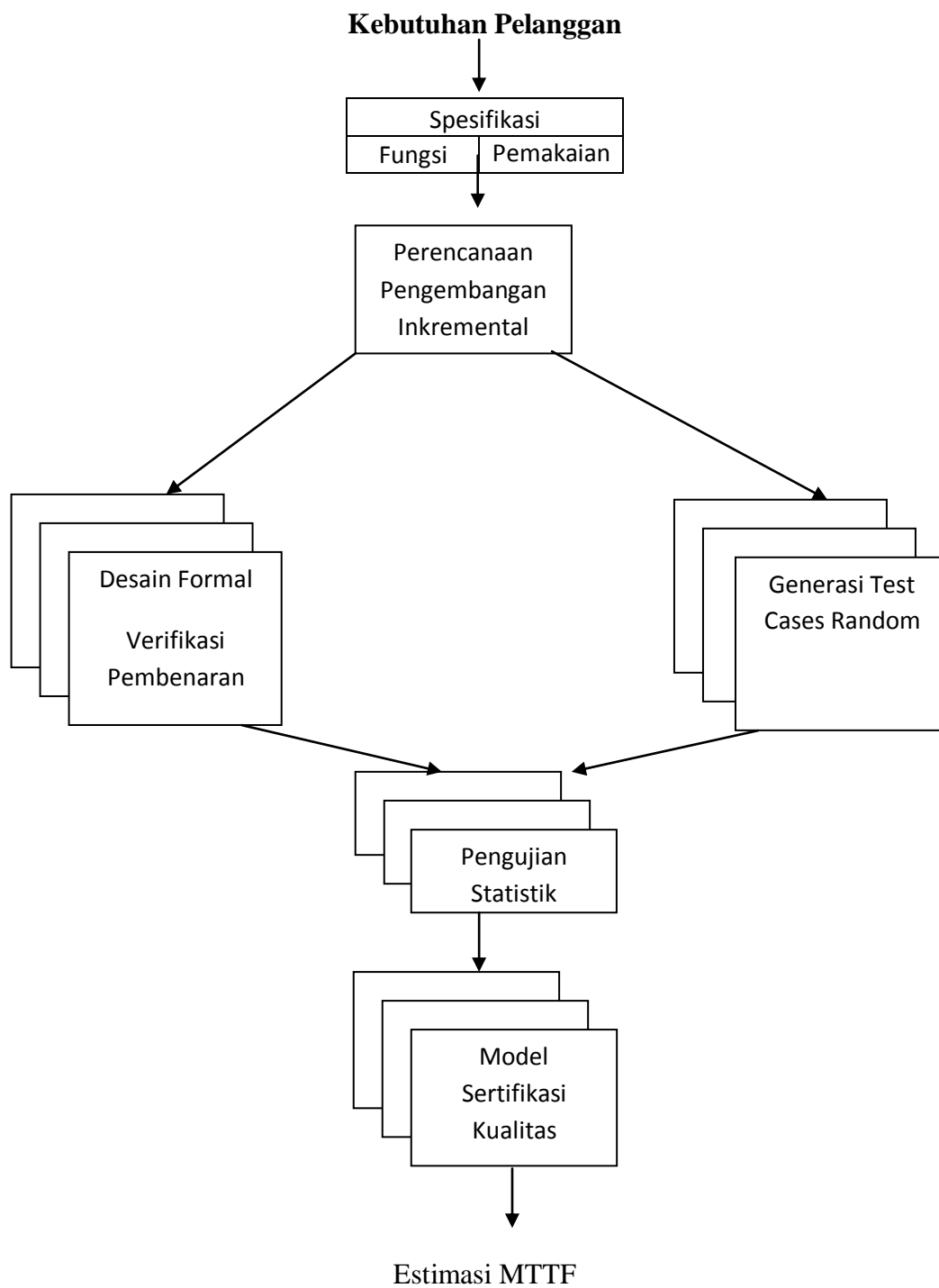
Implementasi dari kelas-kelas telah dikodekan dan unit telah diuji.

Tahap analisa dari proses tersebut terdiri dari langkah 1 dan 2, tahap perancangan terdiri dari langkah 3 sampai 6, dan tahap implementasi terdiri dari langkah 7 dan 8. Beberapa iterasi diharapkan terjadi selama tahap analisa dan perancangan.

2.2.3 Metodologi Cleanroom

Linger dan Hausler (1992), (Rekayasa perangkat lunak *Cleanroom* (*Cleanroom Software Engineering*) dianggap sebagai sebuah proses rekayasa dengan dasar matematika daripada sebuah proses pemrograman coba dan ralat (*trial and error*). Proses *cleanroom* menggunakan teknologi-teknologi berbasis teori, seperti spesifikasi struktur kotak dari fungsi pengguna dan arsitektur objek sistem, perancangan teoritis fungsional verifikasi kebenaran, dan ujian pemakaian statistik untuk sertifikasi kualitas (Simarmata, Janner, 2010 : 86).

Linger (1993), Manajemen Cleanroom berdasar pada pembangunan *incremental* dan sertifikasi dari sebuah garis inkremen (peningkatan) fungsi pengguna (*user-function*) yang bertumpuk sedikit demi sedikit ke dalam produk akhir. Operasi-operasi *Cleanroom* dapat dikeluarkan dengan mudah oleh tim pengembangan independen dan tim sertifikasi (uji) dengan tim dari tim-tim untuk proyek besar. Gambar 2.6 menunjukkan suatu implementasi penuh dari proses *Cleanroom* (Simarmata, Janner, 2010 : 86)..



Sumber : Simarmata, Janner, 2010 : 86

Gambar 2.6 Proses *Cleanroom*

Proses *Cleanroom* menekankan pentingnya tim pengembang untuk mempunyai kontrol intelektual selama proyek berlangsung. Dasar-dasar dari proses tersebut

adalah bukti pembenaran (dari rancangan dan kode) dan sertifikasi kualitas formal melalui pengujian secara statistik.

2.3 Unified Modelling Language (UML)

2.3.1 Pengertian UML

UML muncul Karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak, UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenal sebuah sistem dengan menggunakan diagram dan teks-teks pendukung (S, Rosa A. dan M. Shalahuddin, 2013 : 139).

2.3.2 Diagram UML

2.3.2.1 Use Case Diagram

Use case diagram merupakan pemodelan untuk kelakuan sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang akan ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu (S, Rosa A. dan M. Shalahuddin, 2013 : 155).

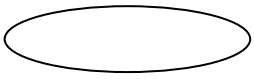
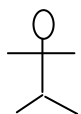

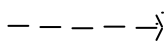
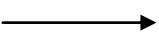
ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut actor dan *use case* (S, Rosa A. dan M. Shalahuddin, 2013 : 155);

- a. *Actor* merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol *actor* belum tentu merupakan orang.

- b. *Use case* merupakan fungsionalitas yang di sediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau actor.

Berikut adalah simbol-simbol yang ada pada *use case* diagram:

Tabel 2.1 Simbol-Simbol *Use Case Diagram*

Simbol	Deskripsi
<i>Use case</i> 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan actor.
Aktor/Actor 	Orang, proses atau sistem lain yang berintegrasi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri.
Asosiasi 	Komunikasi antara actor dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan actor.
Ekstensi 	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambah dapat berdiri walau tanpa <i>use case</i> tambahan itu mirip dengan prinsip <i>inheritance</i> pada pemrograman berorientasi objek.
Generalisasi 	Hubungan generasi dan spesialisasi (umum-khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari yang lainnya.

Menggunakan include/uses <<include>> - - - - -> <<uses>> —————>	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat dijalankan <i>use case</i> ini.
---	--

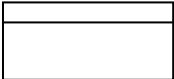


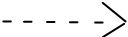

Sumber : (S,Rosa A. dan M. Shalahuddin, 2013: 156)

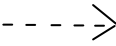
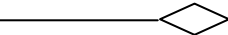
2.3.2.2 Class Diagram

Class diagram menggambarkan struktural item dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem (S, Rosa A. dan M. Shalahuddin,2013 : 141).

Berikut adalah simbol-simbol yang ada pada diagram kelas :

Table 2.2 Simbol-simbol *Class Diagram*.

Simbol	Deskripsi
kelas 	Kelas pada struktural sistem
antarmuka 	Sama dengan konsep interface dalam pemrograman berorientasi objek.
asosiasi 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i>
Asosiasi berarah 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i>
generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)

ketergantungan 	Kebergantungan antar kelas
agregasi 	Relasi antarkelas dengan makna semua bagian (<i>whole part</i>)




Sumber : (S, Rosa A. dan M. Shalahuddin, 2013: 146)

2.3.2.3 State Chart Diagram

State chart diagram digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek. *State chart diagram* merupakan pengembangan dari diagram *finite state automata* dengan perubahan beberapa fitur dan konsep baru (S, Rosa A. dan M. Shalahuddin, 2013 : 163).

Berikut ini komponen-komponen yang sering digunakan pada saat pembuatan *state machine diagram* (S, Rosa A. dan M. Shalahuddin, 2013 : 163) :

Tabel 2.3 Simbol-Simbol State Machine Diagram

Simbol	Deskripsi
<i>Start</i> 	<i>Start</i> atau <i>initial state</i> adalah <i>state</i> atau keadaan awal pada saat sistem mulai hidup.
<i>End</i> 	<i>End</i> atau <i>final state</i> adalah <i>state</i> keadaan akhir dari daur hidup suatu sistem.
<i>Even</i> 	<i>Even</i> adalah kegiatan yang menyebabkan perubahan status mesin
<i>State</i>	Sistem pada waktu tertentu






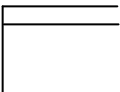
Sumber : (S , Rosa A. dan M. Shalahuddin, 2013:164)

2.3.2.4 Activity Diagram

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram *activity* menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (S, Rosa A. dan M. Shalahuddin, 2013 : 161).

Berikut ini simbol-simbol yang sering digunakan pada saat pembuatan *activity diagram*:

Tabel 2.4 Simbol-Simbol Activity Diagram

Simbol	Deskripsi
Status awal 	Status awal aktifitas sistem, sebuah diagram aktifitas memiliki sebuah status awal
aktivitas 	Aktifitas yang dilakukan sistem, aktifitas biasanya diawali dengan kata kerja
Percabangan 	Asosiasi percabangan dimana jika ada pilihan aktifitas lebih dari satu
Penggabungan 	Asosiasi penggabungan dimana lebih dari satu aktifitas digabung menjadi satu.
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktifitas memiliki sebuah status akhir.
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang ada.

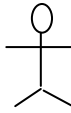



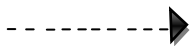
Sumber : (S, Rosa A. dan M. Shalahuddin, 2013: 162)

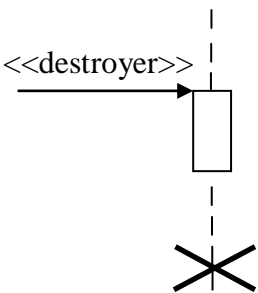
2.3.2.5 Sequence Diagram

Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek (S, Rosa A. dan M. Shalahuddin : 165).

Berikut ini simbol-simbol yang sering digunakan pada saat pembuatan *sequence diagram*:

Tabel 2.5 Simbol-Simbol *Sequence Diagram*

Simbol	Deskripsi
<p>Actor</p>  <p>Nama <i>actor</i> Atau</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Nama <i>actor</i></div> <p>Tanpa waktu aktif</p>	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem.
<p>Garis hidup</p> <p>-----</p>	Menyatakan kehidupan suatu objek
<p>objek</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">_____</div>	Merupakan objek yang berinteraksi pesan
<p>Waktu aktif</p> <div style="border: 1px solid black; width: 20px; height: 30px; display: inline-block;"></div>	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan.
<p>Pesan tipe <i>create</i></p> <p><<create>></p> 	Menyatakan suatu objek memuat objek yang lain, arah panah mengarah pada objek yang dibuat.
<p>Pesan tipe <i>call</i></p> <p>1: nama_metode()</p> 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.
<p>Pesan tipe <i>send</i></p> <p>1: masukan</p> 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya
<p>Pesan tipe <i>return</i></p> <p>1: keluaran</p> 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembali ke objek tertentu

<p>Pesan tipe <i>destroy</i></p> 	<p>Menyatakan suatu objek mengakhiri hidup objek yang lain</p>
--	--

Sumber : (S, Rosa A. dan M. Shalahuddin, 2013 : 165)

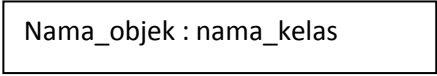

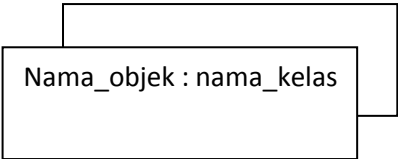

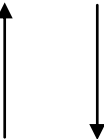
2.3.2.7 Collaboration Diagram

Collaboration diagram menggambarkan interaksi antar objek/bagian dalam bentuk urutan pengiriman pesan. *Collaboration* diagram merepresentasikan informasi yang diperoleh dari diagram kelas, diagram sekuen, dan diagram *use case* untuk mendeskripsikan gabungan antara struktural statis dan tingkah laku dinamis dari suatu sistem. *Collaboration* diagram adalah bentuk lain dari *sequence* diagram. Bila *sequence* diagram diorganisir menurut waktu maka *collaboration* diagram diorganisasi menurut ruang dan waktu (S, Rosa A. dan M. Shalahuddin, 2013 : 168).

Collaboration diagram mengelompokkan message pada kumpulan diagram sekuen menjadi sebuah diagram. Dalam diagram *collaboration* yang dituliskan adalah operasi/metode yang dijalankan antara objek yang satu dan objek lainnya secara keseluruhan, oleh karena itu dapat diambil dari jalannya interaksi pada semua diagram sekuen. Penomoran metode dapat dilakukan berdasarkan urutan dijalankannya metode/operasi diantara objek yang satu dengan objek lainnya atau objek itu sendiri (S, Rosa A. dan M. Shalahuddin, 2013 : 168).

Berikut adalah simbol-simbol yang ada pada diagram kolaborasi :

Tabel 2.6 Simbol-Simbol *Collaboratio Diagram*

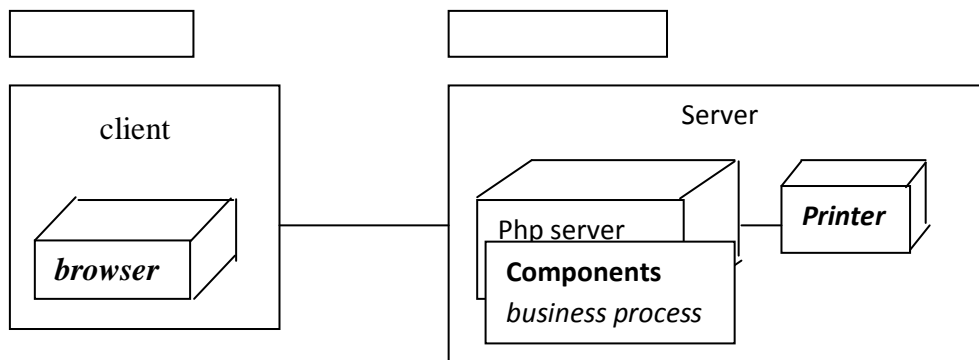
Simbol	Deskripsi
<p>Objek</p> 	<p>Objek yang melakukan interaksi pesan</p>
<p><i>Links</i></p> 	<p>Relasi antar objek yang menghubungkan objek satu dengan lainnya atau dengan dirinya sendiri</p> 
<p>Arah pesan</p> 	<p>Arah pesan yang terjadi, jika pada suatu link ada dua arah pesan yang berbeda maka arah juga digambarkan dua arah pada dua sisi link</p> 

Sumber : (S, Rosa A. dan M. Shalahuddin, 2013 :168)

2.3.2.7 *Deployment Diagram*

Diagram *deployment* atau *deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal berikut (S, Rosa A. dan M. Shalahuddin : 154):

- Sistem tambahan (*embedded sistem*) yang menggambarkan rancangan *device*, *node*, dan *hardware*
- Sistem *client/server* misalnya seperti gambar berikut :

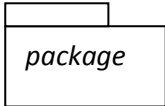
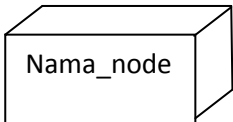




c. Sistem terdistribusi murni

d. Rekayasa ulang aplikasi

Berikut adalah simbol-simbol yang ada pada diagram *deployment*:

Tabel 2.7 Simbol-Simbol *Deployment Diagram*

Simbol	Deskripsi
<i>Package</i>  <i>package</i>	Package merupakan sebuah bungkus dari satu atau lebih <i>node</i>
<i>Node</i>  Nama_node	Biasa mengacu pada perangkat keras (<i>hardware</i>), perangkat lunak yang tidak dibuat sendiri (<i>software</i>), jika didalam node disertakan komponen untuk mengkonsitensikan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen.
Ketergantungan / <i>dependency</i> 	Ketergantungan antar komponen, arah panah mengarah kepada komponen yang dipakai.
Antarmuka/ <i>interface</i>  Nama_interface	Sama dengan konsep <i>interface</i> pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen.

<i>link</i> _____	Relasi antar komponen
--------------------------	-----------------------

Sumber : *S, Rosa A. Dan M. Shalahuddin (2013 : 154)*

2.5 Media Pembelajaran

Menurut David, Bern (1991) pada penelitian Fahaluddin, Iwan (2014) tentang *Pemanfaatan Media dalam Pembelajaran* mengemukakan bahwa pada mulanya media pembelajaran hanyalah dianggap sebagai alat untuk membantu pembelajar dalam proses kegiatan mengajar (*teaching aids*). Alat bantu mengajar berikutnya adalah alat bantu visual seperti gambar, model, grafis, atau benda nyata lain. Alat-alat bantu itu dimaksudkan untuk memberikan pengalaman lebih konkret, baik, memotivasi serta mempertinggi daya serap dan daya ingat pembelajar dalam belajar .

Menurut P. Wiratmojo dan Sasonohardjo (2002), Pada penelitian Fahaluddin, Iwan (2014) tentang *Pemanfaatan Media dalam Pembelajaran* mengemukakan bahwa Media pembelajaran merupakan alat peraga dengan alat bantu yang digunakan oleh pembelajar untuk mempermudah tugas dalam mengajar dengan memperagakan fakta, konsep, prinsip, atau prosedur tertentu .

2.6 Virus

Virus dikatakan benda mati, tetapi jika mendapatkan tempat pada sel hidup/organisme, virus akan menunjukkan aktivitas seperti sel hidup, yaitu mampu berkembangbiak sehingga jumlah bertambah. Dengan demikian virus adalah peralihan antara benda mati dan makhluk hidup. Dikatakan peralihan karena memiliki ciri-ciri seperti makhluk hidup yaitu mempunyai DNA dan dapat

berkembangbiak pada sel hidup serta memiliki ciri-ciri benda mati yaitu tidak mempunyai protoplasma dan mampu dikristalkan (Subardi, dkk, 2009 : 27).

Sebagian beras virus membawa sekitar 50 gen di dalam selubung proteinnya, meskipun beberapa virus hanya memiliki tiga gen serta ada pula yang 300 gen. Virus merupakan penyebab beberapa timbulnya penyakit pada manusia, hewan maupun tumbuhan (Subardi, dkk, 2009 : 27).

Ciri-ciri virus (Subardi, dkk, 2009 : 27):

1. Tidak memiliki bentuk sel (aseluler)
2. Berukuran antara (20-300) milimikron
3. Hanya memiliki satu macam asam nukleat saja yaitu ADN (asam siosiribo nukleat) atau ARN (asam ribo nukleat)
4. Berupa hablur atau kristal dengan bentuk bervariasi, berikut bentuk-bentuknya:
 - a. Oval
 - b. Memanjang
 - c. Silindris
 - d. Kotak, dan lain-lainnya.
5. Tubuhnya tersusun atas kepala, kulit selubung (kapsid) yang berisi ADN atau ARN dan serabut ekor.

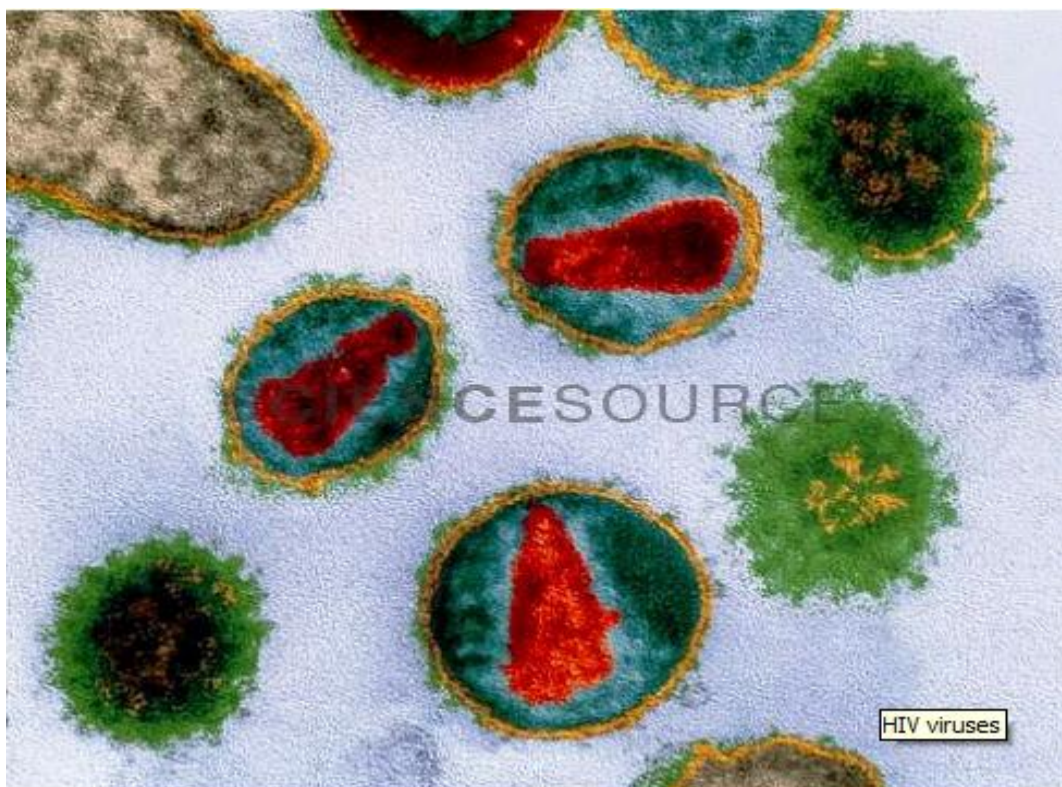
Macam – macam virus pada manusia:

1. Virus HIV

Virus HIV adalah virus yang menyerang kekebalan tubuh pada manusia sehingga terjadi penurunan kekebalan tubuh pada orang yang menderita. Virus HIV terdapat pada penderita AIDS, penyebab masuknya virus HIV kedalam tubuh melalui

hubungan seksual, jarum suntik yang tidak steril, transfusi darah dan ibu yang menderita penyakit AIDS kepada anak yang sedang dikandungnya (Subardi, dkk, 2009 : 27).

Gambar 2.7 adalah foto dari *virus* HIV



Sumber : www.sciencesource.com

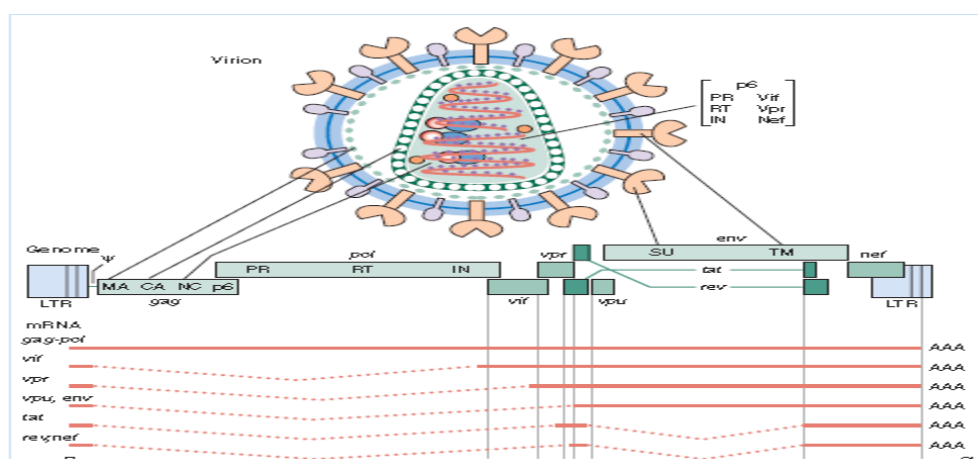
Gambar 2.7 Virus HIV

Etiologi HIV-AIDS adalah Human Immunodefisiensi virus (HIV) yang merupakan virus sitopatik yang diklasifikasikan dalam famili retroviridae, subfamili lentiviridae, genus lentivirus. Berdasarkan strukturnya HIV termasuk famili tetrovirus yang merupakan kelompok virus RNA yang mempunyai berat molekul 0,7 kb (kilobase). virus ini terdiri dari 2 grup, yaitu HIV-1 dan HIV-2. Masing-masing grup mempunyai sub tipe. Diantara kedua grup itu, yang paling banyak menimbulkan kelainan dan lebih ganas di seluruh dunia adalah grup HIV-

1 (United States Preventive Services Task Force, 2011) (Yuliyanasari, Nurma. 2017).

.HIV terdiri dari bagian inti berbentuk silindris yang dikelilingi oleh *lipid bilayer envelope*. Pada *lipid bilayer* tersebut terdapat dua jenis glikoprotein yaitu gp120 dan gp41. fungsi utama protein ini adalah untuk memediasi pengenalan sel CD4+ dan reseptor kemokin dan memungkinkan virus untuk melekat pada sel CD4+ yang terinfeksi. Bagian dalam terdapat dua kopi RNA juga berbagai protein dan enzim yang penting untuk replikasi dan maturasi HIV antara lain adalah p24, p7, p9, p17, reverse transkriptas, integras, dan protease. Tidak seperti retrovirus yang lain, HIV menggunakan sembilan gen untuk mengkode protein penting dan enzim. Ada tiga gen utama yaitu *gag*, *pol*, dan *env* mengkode komponen struktural HIV yaitu glikoprotein. Sementara itu gen *rev*, *nef*, *vif*, *vpu*, *vpr*, dan *tat* penting untuk replikasi virus dan meningkatkan tingkat infeksi HIV (Calles, et al. 2006, Kummar, et al. 2015)

Struktur virus HIV (Yuliyanasari, Nurma. 2017) :

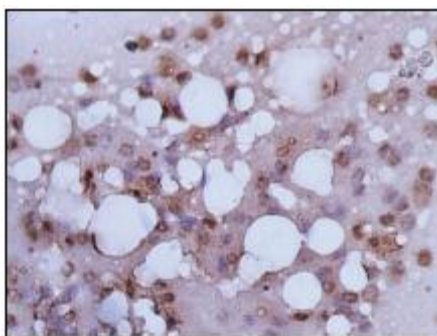


Sumber : Edwar K. Wagner, et al. 2006 : 401

Gambar 2.8 Struktur Virus HIV

2. Virus Dengue

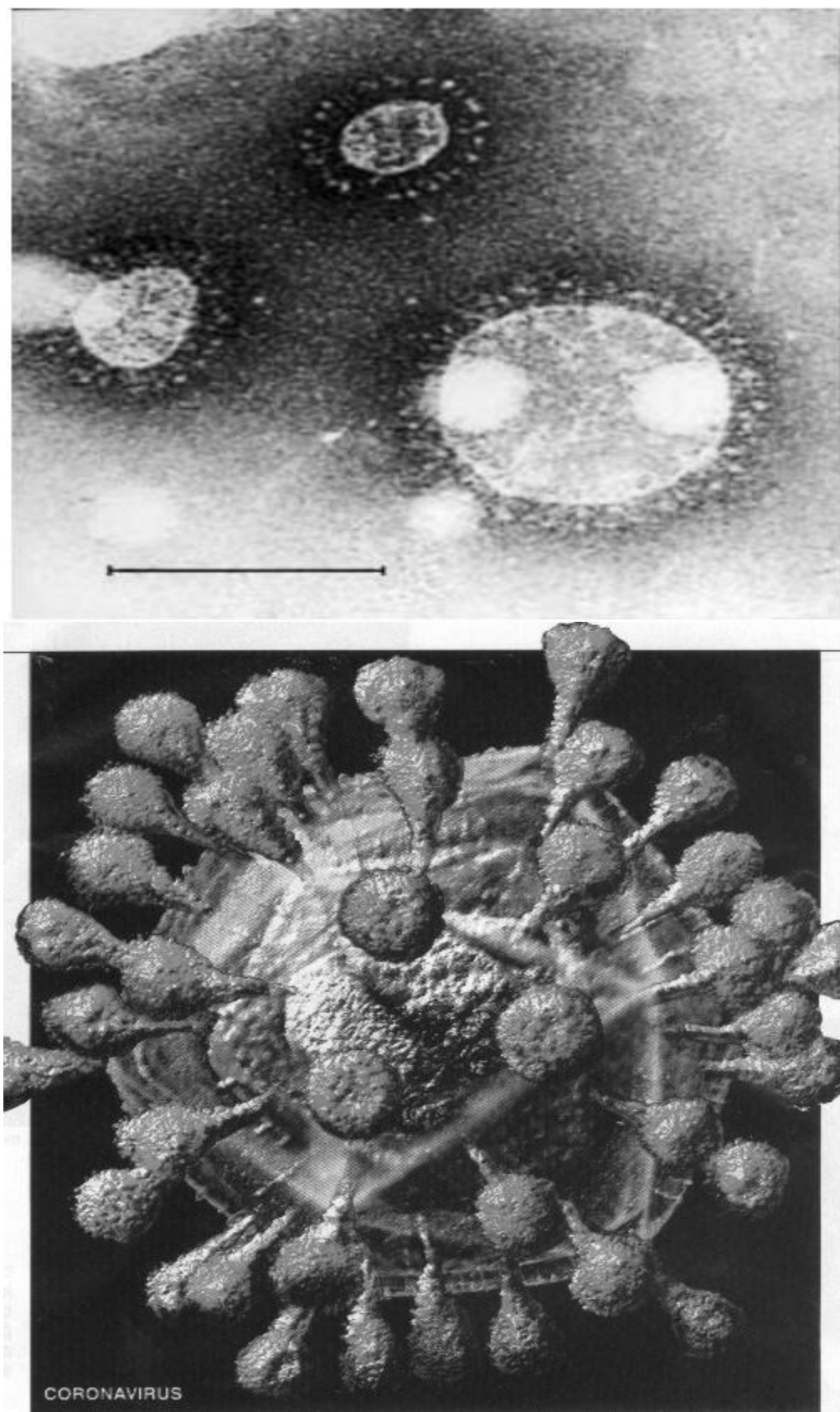
Virus dengue adalah virus yang dapat menyebabkan kadar trombosit dan menyebabkan pecahnya kapiler darah sehingga gejala-gejala yang tampak adalah adanya bercak-bercak merah pada kulit, demam panas tinggi, sakit kepala, mimisan lebih parah lagi pendarahan pada organ-organ tubuh dan dapat menyebabkan kematian. Virus dengue terdapat pada penderita Demam Berdarah (DB). Faktor penyebab penyakit ini adalah nyamuk *Aedes aegypti* (Rampengan, Novie H, 2016)..



Gambar 2.9 Virus Dengue

3. Virus Corona

Virus corona adalah virus yang dapat menyebabkan suhu tubuh diatas 40 derajat C, menggigil, kelelahan otot, batuk kering, sakit kepala, sesak nafas, dan diare. Virus corona terdapat pada penderita SARS (*Severe Acute Respiratory Syndrome*). Virus ini dibawa oleh mamalia golongan musang dan rakun. Virus ini mudah sekali mengalami mutasi [4]. Gambar 2. adalah virus corona dari elektron mikroskop yang termasuk anggota keluarga *Filoviridae* dengan skala bar = 100 nm (W. J. Mahy, Brian., & H. V. Van Regenmortel, Marc. 2010).



Sumber : *Brian W. J. Mahy & Marc H. V. Van Regenmortel, 2010:508*

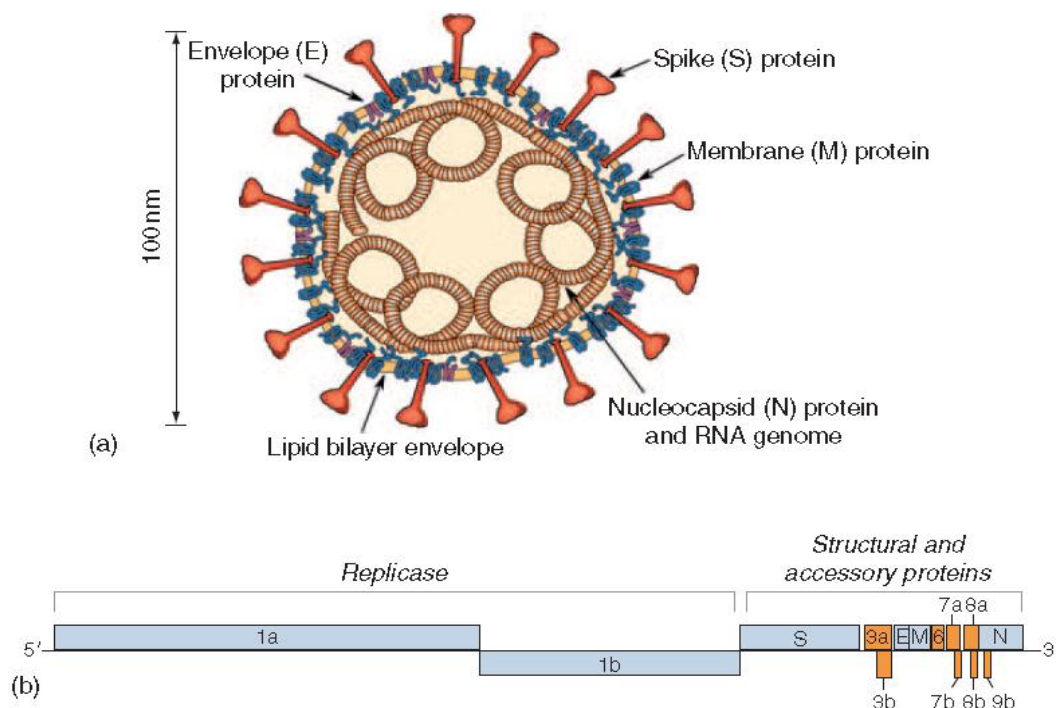
Gambar 2.10 *Virus Corona*

Awalnya nama virus ini *Human Coronavirus-EC*, tapi kemudian oleh konsesus global berubah menjadi MERS-CoV. Virus ini merupakan versi beta Corona Virus garis keturunan C. Struktur dari MERS-CoV menggambarkan dipeptid-pepsidae 4 (DPP4 atau CD26) diidentifikasi sebagai reseptor host-sel untuk *entry* sel.

Saat ini ada beberapa Corona virus yang dapat mempengaruhi manusia yaitu:

1. Corona virus 229E manusia dan Corona virus NL63 manusia
2. Corona virus OC43 manusia, Corona virus HKU1 manusia, SARS-CoV, dan MERS-Cov.

Corona Virus biasanya menginfeksi satu jenis spesies atau yang terkait erat, seperti melalui spesimen hidung unta. Kemungkinan penularan MERS dapat melalui kontak langsung dari percikan dahak dan tidak langsung melalui kontak benda yang terkontaminasi virus. Struktur dari virus corona (Granoff, Allan., & Webster, Robert G. 2010).



Sumber : Granoff, Allan., & Webster, Robert G. 2010 : 508

Gambar 2.11 Struktur virus corona

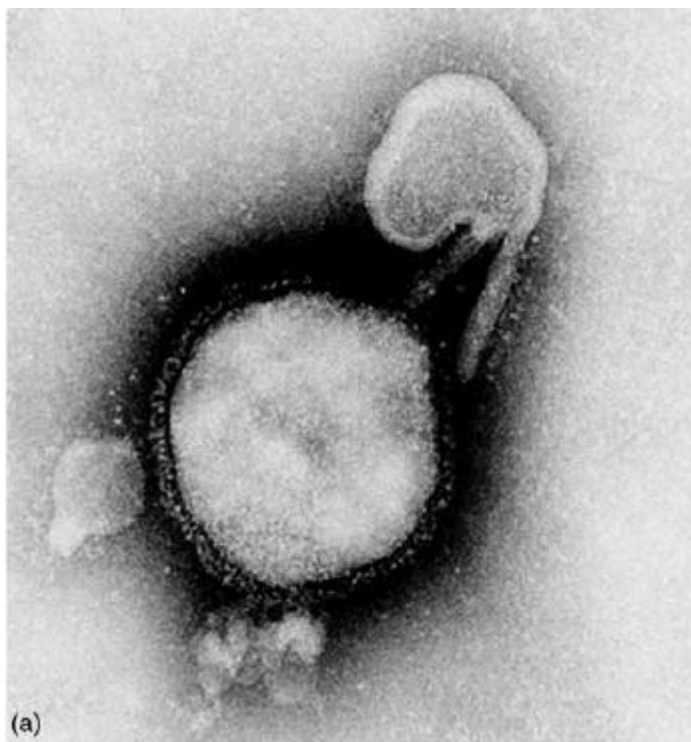
Vaksin dari MERS belum ditemukan obat yang bekerja melawan MERS tersebut, akan tetapi tindakan pencegahan dapat dilakukan dengan perilaku hidup bersih dan sehat, menghindari kontak erat dengan penderita, menggunakan masker, menjaga kebersihan tangan dengan sering mencuci tangan memakai sabun dan menerapkan etika batuk ketika sakit. Pencegahannya juga sama seperti pencegahan infeksi pada penyakit flu burung dan *Emerging infectious Disease* lainnya yang mengenai saluran napas (Rampengan, Novie H, 2016).

4. Virus Orthomyxo

Virus orthomyxo adalah virus yang menyebabkan penderita mengalami kesulitan bernapas. Virus ini menyerang saluran pernapasan. Gejala-gejalanya seperti demam, sakit kepala, pegal linu, kehilangan nafsu makan. Virus orthomyxo terdapat pada penderita Influenza. Penyebaran dari virus ini melalui udara yang terserap masuk melalui saluran pernapasan (Subardi, dkk, 2009 : 30).

5. Virus Paramyxo

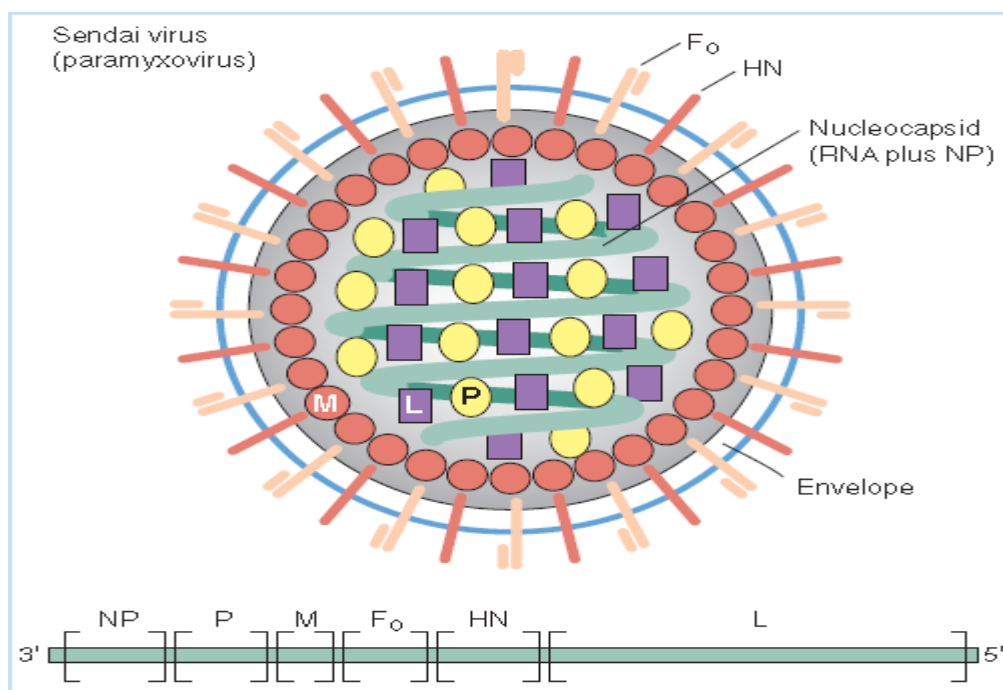
Virus paramyxo adalah virus yang menyebabkan demam tinggi, batuk, mata pedih jika terkena cahaya dan rasa ngilu di semua tubuh. virus paramyxo terdapat pada penderita campak (morbili). Di awal masa inkubasi virus berkembangbiak di saluran pernapasan atas. Diakhir masa inkubasi virus menuju ke darah dan beredar ke seluruh bagian tubuh terutama kulit. Gambar 2.1 adalah virus paramyxo dari mikroskop elektron (W. J. Mahy, Brian., & H. V. Van Regenmortel, Marc. 2010).



Sumber : W. J. Mahy, Brian., & H. V. Van Regenmortel. 2010 : 500

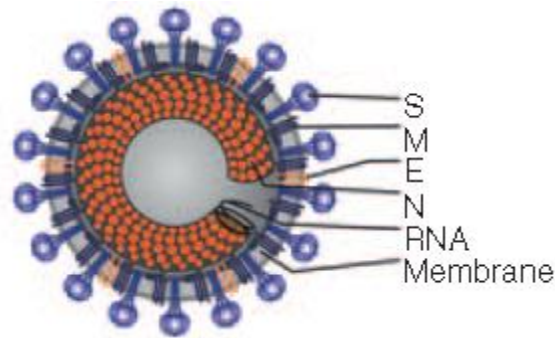
Gambar 2.12 Virus Paramyxo

Struktur dari virus paramyxo (Edwar K. Wagner, et al. 2006) :



Sumber : Edwar K. Wagner, et al. 2006 : 281

Gambar 2.13 Struktur Virus Paramyxo



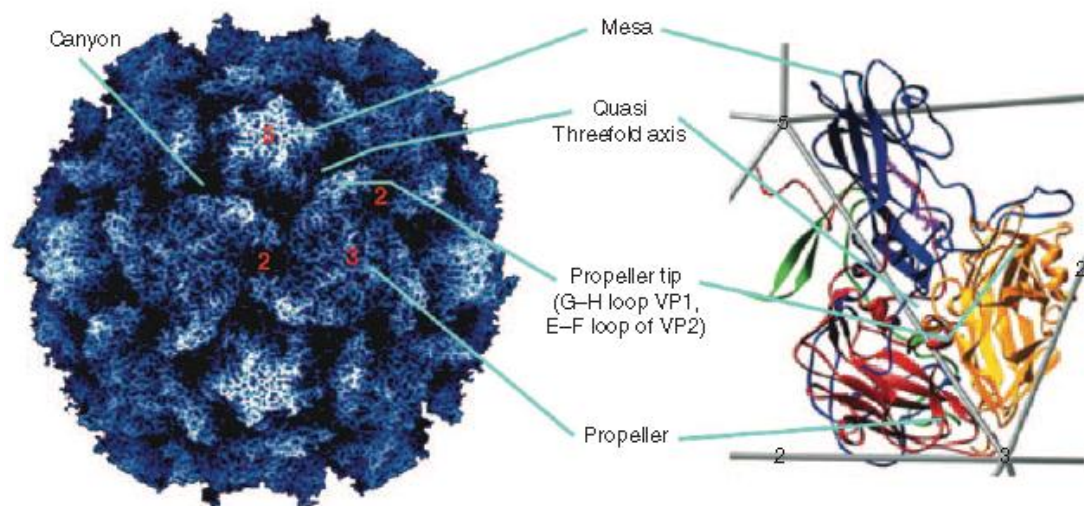
Sumber : *Granoff, Allan., & Webster, Robert G. 2010 : 508*

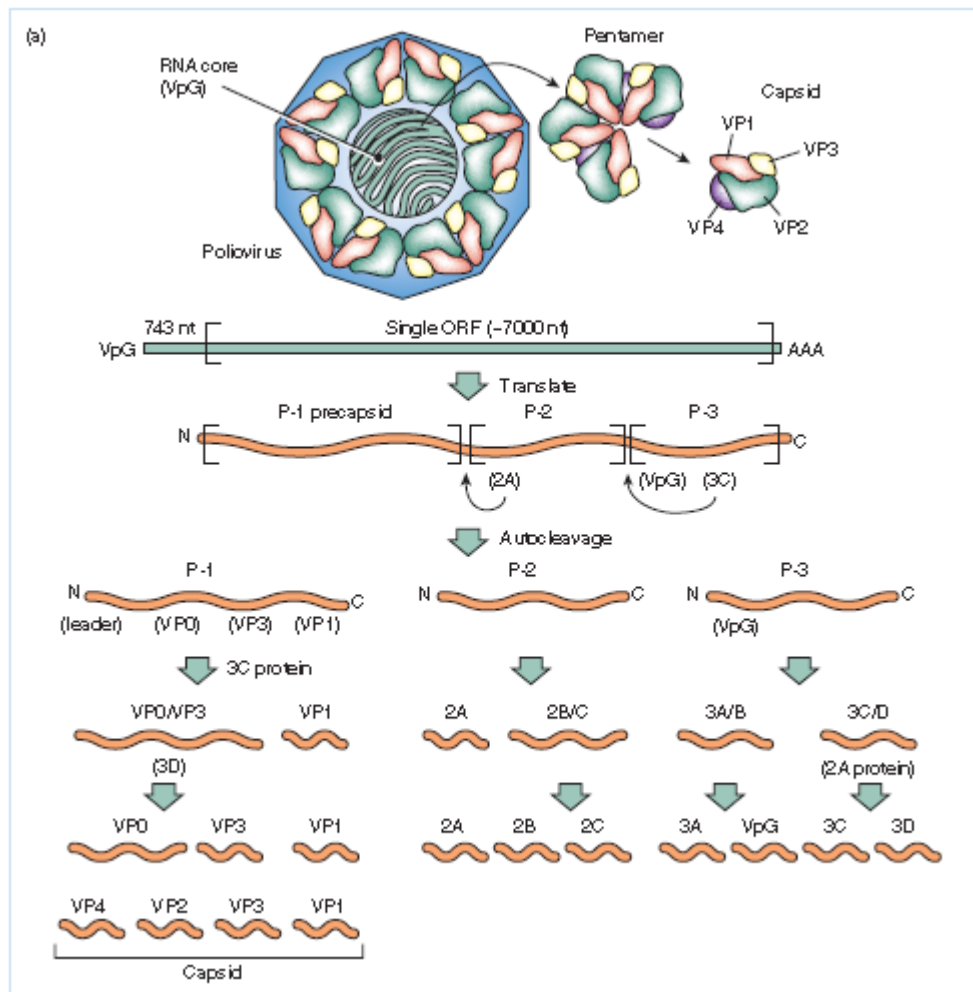
Gambar 2.14 Struktur Virus Paramxyo

6. Virus Polio

Virus polio menyerang pada anak-anak. Virus ini adalah virus yang dapat menyebabkan kelumpuhan bila menyerang selaput meninges otak dan merusak sel saraf di depan otak. Virus polio terdapat pada penderita polio. Gejala dari penderita antara lain; demam, sakit kepala, tidak enak badan, mengantuk, sakit tenggorokan, mual dan muntah. Vaksin untuk polio adalah vaksin Salk dan Sabin. Vaksin Salk berfungsi mengaktifkan produksi antibodi di serum, menetralkan virus yang virulan saat memasuki aliran darah dan mencegah serangan ke sistem saraf pusat. Sedangkan vaksin sabin mengandung virus polio yang telah dilemahkan.

Struktur dari virus polyo



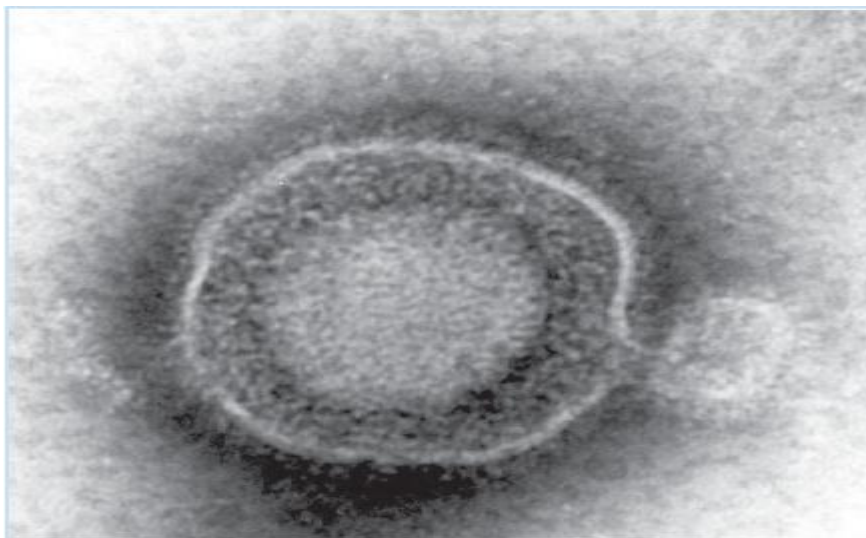


Sumber : Edwar K. Wagner, et al. 2008:250

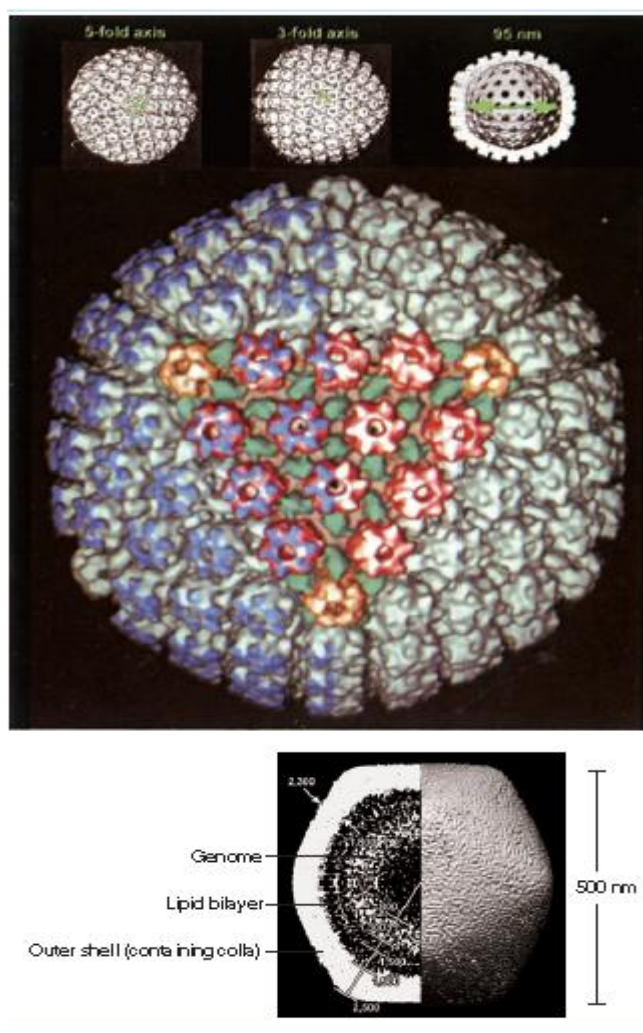
Gambar 2.15 Struktur Virus Polio

7. Virus *Herpesvirusvaricellae*

Virus *herpesvirusvaricellae* terdapat pada penderita cacar. Virus ini menyebabkan bentol-bentol disertai nanah yang menyerang permukaan kulit. Jika sembuh akan meninggalkan bekas luka pada kulit tubuh dan wajah.



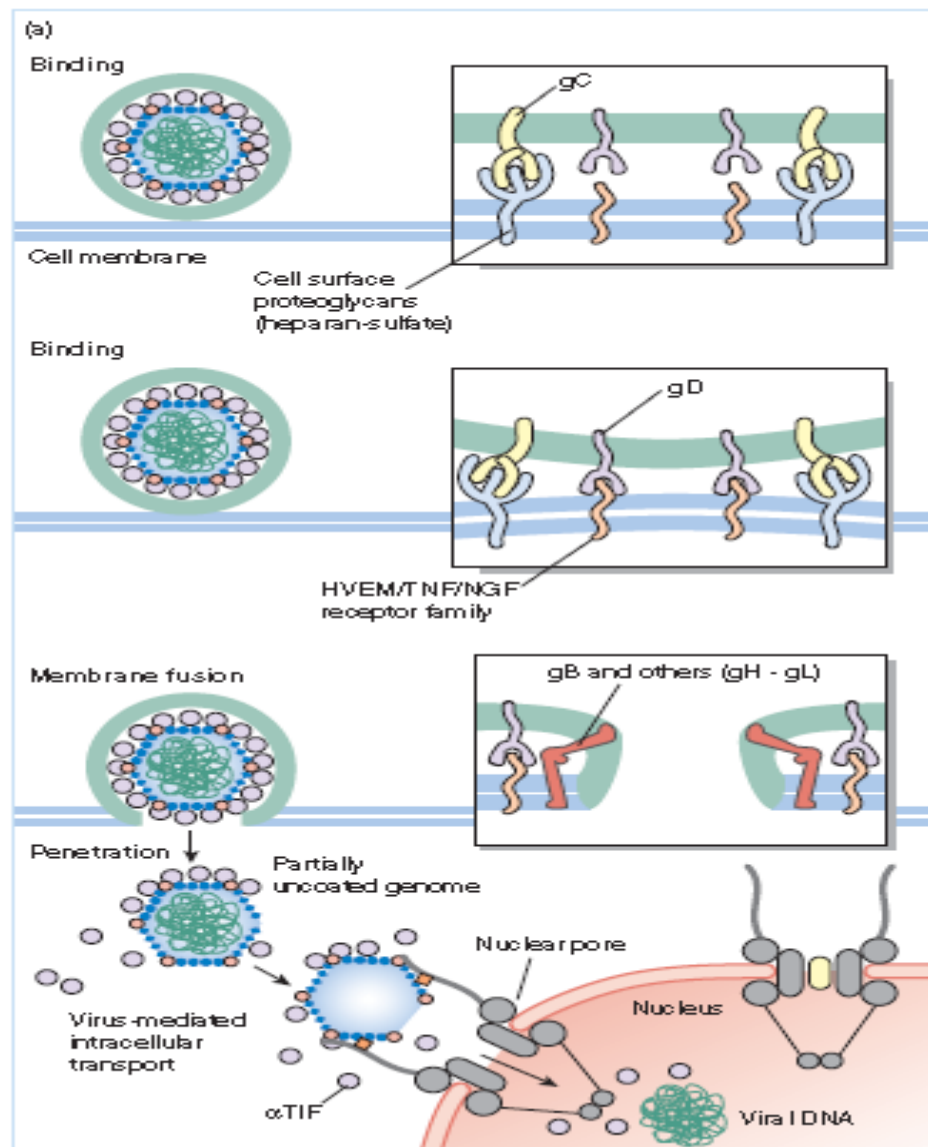
Gambar 2.16 Struktur *Virus Herpes*



Sumber : *Edwar K. Wagner, et al. 2008:260*

Gambar 2.17 Struktur *Virus Herpes*

Masuknya HSV-1 ke dalam sel (*Edwar K. Wagner, et al, 2006*) :



Sumber : *Edwar K. Wagner, et al. 2008:340*

Gambar 2.18 Masuknya HSV-1 ke dalam sel

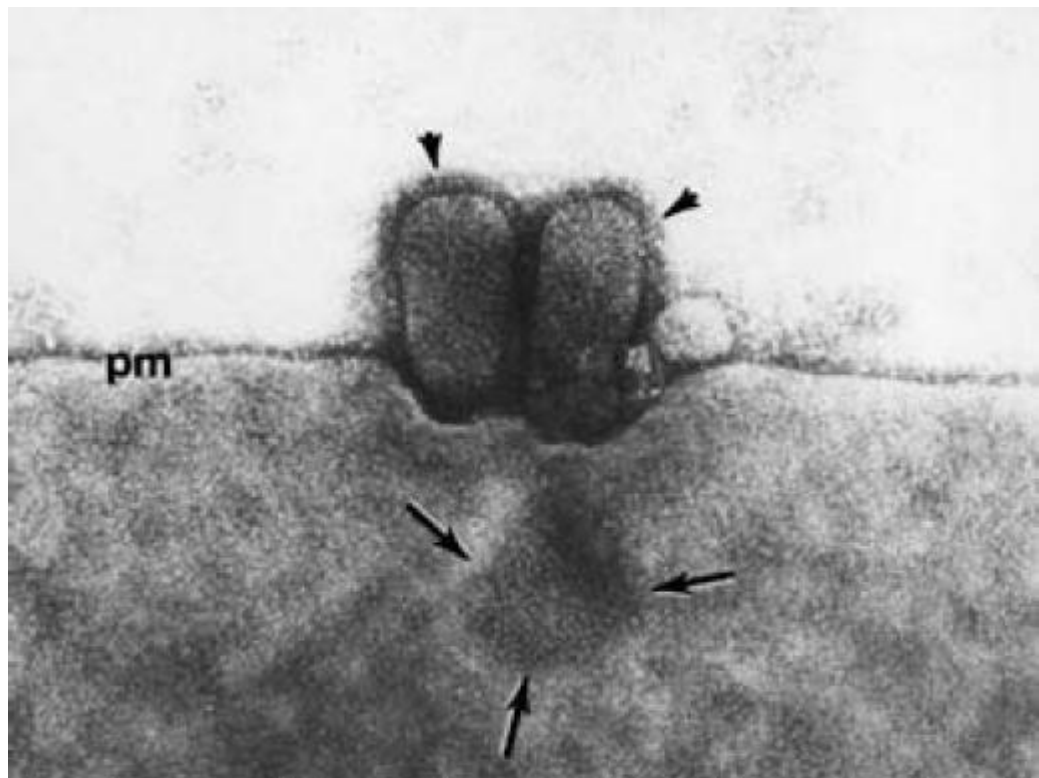
Gambar 2.18 Masuknya HSV-1 ke dalam sel untuk inisiasi infeksi. (a) Garis besar proses. Asosiasi awal adalah antara proteoglikan dari permukaan dan glikoprotein C (gC); Hal ini diikuti oleh interaksi spesifik dengan salah satu beberapa reseptor seluler secara kolektif menyebut entri herpesvirus mediator (HVEMs). Ini berhubungan dengan reseptor saraf faktor pertumbuhan (NGFs) dan tumor necrosis factor (TNF). Itu asosiasi membutuhkan interaksi spesifik dengan glikoprotein D (gD).

Fusion dengan membran seluler berikut; ini memerlukan tindakan sejumlah virus glikoprotein termasuk gB, gH, gI, dan gL. Sebuah studi mikroskop elektron Fusi herpesvirus dengan sel yang terinfeksi ditunjukkan pada Gambar.6.3. Kapsul virus dengan beberapa protein tegument itu bermigrasi ke pori-pori nuklir yang memanfaatkan mesin transportasi seluler. "Docking" ini diperkirakan menghasilkan DNA virus disuntikkan melalui pori-pori sementara kapsid tetap berada di sitoplasma. Beberapa protein tegument, seperti -TIF, juga masuk ke nukleus dengan genom virus. (b) mikroskop elektron analisis virus kapsul pseudorabies "docking" dan genom injeksi di pori-pori nuklir. Urutan logis ditampilkan maju dari kapsul (gelap) sampai yang kosong. Itu prosesnya sangat mirip dengan injeksi DNA bakteriofag ke dalam sel bakteri.

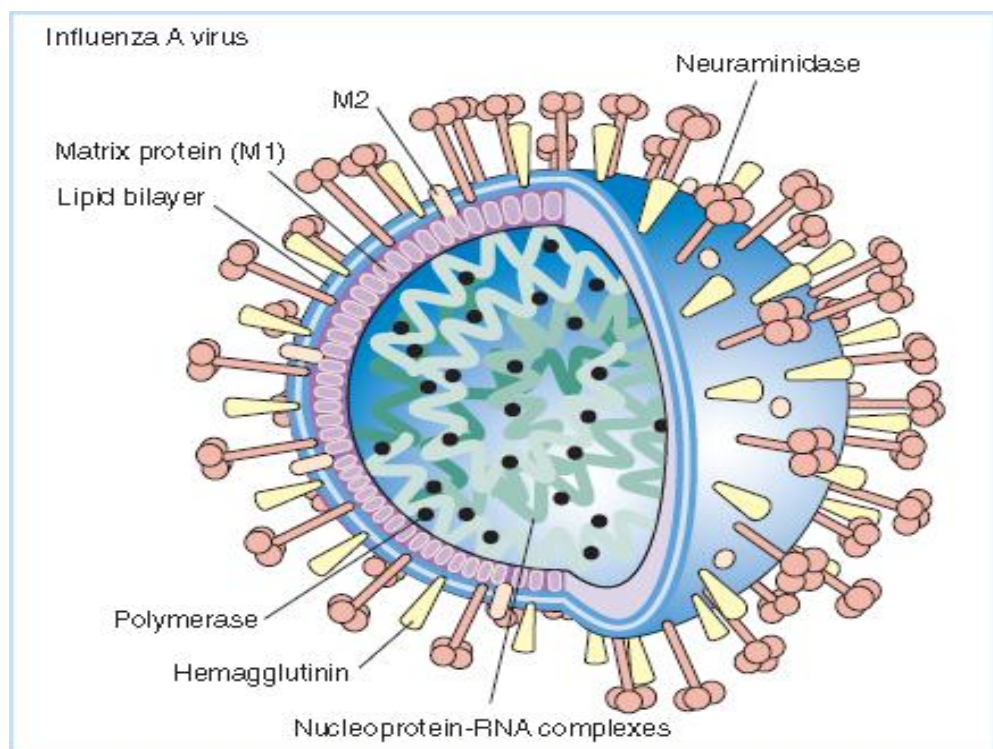
8. Virus Avian Influenza

Virus avian influenza (VAI) merupakan virus influenza A terdiri atas 8 potongan RNA berpolimorfik negatif dan termasuk dalam famili *Orthomyxoviridae*. Virus ini pada permukaannya di selimuti oleh sekitar 500 glikoprotein. Kedelapan potongan RNA virus tersebut berukuran 13,5 kilobasa (kb) yang diselimuti oleh protein nukleokapsid, dengan panjang segmen berkisar antara 890 sampai dengan 2341 nukleotida dan terdiri dari 23-45 nukleotida non coding pada ujung 3' dan 23-61 nukleotida pada ujung 5'. Tiap-tiap segmen yang ada akan mengkode suatu protein fungsional yang penting yang terdiri atas protein polimerase A (PA), Hemagglutinin (HA), Protein nukleokapsid, Neuraminidase (NA), Protein Matriks (M) dan protein non-struktural (NS). Dari seluruh komponen yang ada, kemudian akan bersama-sama akan membentuk ribonukleoprotein (RNP) (Werner & Harder, 2006; Gurtler, 2006; Ghedin, et al., 2005; WHO, 2002; Muramoto, et al., 2006; UGM, 2005) (Garjito, Tribowo Ambar, 2013).

struktur virus Influenza A (*Edwar K. Wagner, et al, 2006*):



Gambar 2.19 Virus Avian Influenza



Sumber :*Edwar K. Wagner, et al. 2008:284*

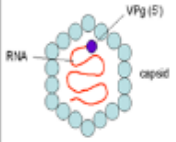
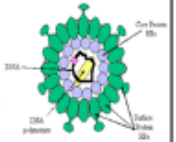
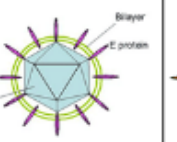
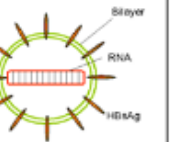

Gambar 2.20 Struktur Virus Avian Influenza

Virus Avian Influenza tertular dari unggas melalui kontak langsung maupun tidak langsung dengan resiko kematian yang relatif tinggi. Upaya untuk mencegah maupun pengobatan telah dilakukan oleh berbagai pihak, meskipun demikian keberhasilan terapi umumnya masih jauh dari yang diharapkan (Dwiprahasto, 2006).

9. Virus Hepatitis

Virus hepatitis C merupakan penyebab kedua epidemi infeksi virus setelah *human immunodeficiency virus* (HIV). Infeksi hepatitis C umumnya ditemukan pada penderita HIV karena kedua virus tersebut memiliki kesamaan rute transmisi (Mohsen AH, Easterbook P, Taylor CB, dan Norris S, 2002) (Kurniawati, Sri Agustina., Karjadi, Teguh H., & Gani, Rino A, 2015).

Hepatitis C mudah ditularkan melalui rute parenteral seperti penggunaan narkoba suntik dan transfusi darah, akan tetapi sulit ditularkan melalui rute seksual [7]. berbagai faktor diduga berperan dalam meningkatkan risiko transmisi seksual hepatitis C pada penderita koinfeksi HIV/HCV. Tingginya kadar HCV RNA darah HCV dan rendahnya hitung CD4⁺ pasien koinfeksi HIV/HCV (Kurniawati, Sri Agustina., Karjadi, Teguh H., & Gani, Rino A, 2015).

					
Name of Virus	Hepatitis A Virus (HAV)	Hepatitis B Virus (HBV)	Hepatitis C Virus (HCV)	Hepatitis D Virus (HDV)	Hepatitis E Virus (HEV)
Classification	Picornavirus	Hepadnavirus	Flavivirus	Deltavirus	Hepevirus
Viral genome	ssRNA	dsDNA	ssRNA	-ssRNA (-ve)	ssRNA
Transmission	Enteric	Parental	Parental	Parental	Enteric
Incubation period	15-45 days	45-160 days	15-150 days	30-60 days	15-60 days
Chronic Hepatitis	No.	Yes. 10% chance	Yes. >50% chance	Yes. <5% of coinfectious >80% of superinfectious	No.
Cure?	No cure. Treatments usually tackle the symptoms.	No cure. Treatments usually tackle the symptoms.	No cure. Treatments usually tackle the symptoms.	No cure. Treatment: Alpha interferon for 12 months.	No cure. Treatments usually tackle the symptoms

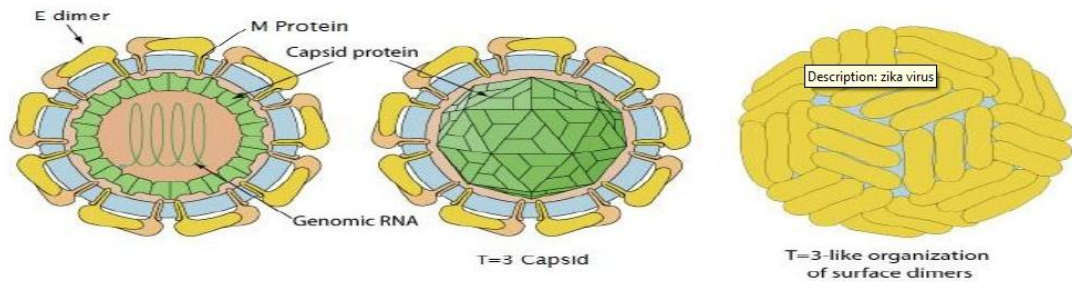
Gambar 2.21 Struktur Virus Hepatitis A, B, C, D, dan E

10. Virus Zika

Virus zika (ZIKV) merupakan golongan genus *Flavivirus*; seperti flavivirus lainnya, virus zika adalah virus RNA *single-standed* yang terbungkus ikosahedral yaitu kumpulan lipid dilindungi oleh tonjolan padat yang terdiri membran dan glikoprotein.

Infeksi virus zika menyebabkan penyakit *self-limited* yang ringan. Masa inkubasinya sekitar 3-12 hari, karena sifatnya yang ringan infeksi dari virus zika mungkin tidak tahu. Spektrum penyakit virus zika tumpang tindih dengan lainnya yaitu infeksi arboviral, tapi beruam (makulopapular dan mungkin termediasi imun).

Virion zika berbentuk ikosahedral. Terbungkus dengan diameter 18-45 nanometer. Genomenya adalah RNA strand positif yang diselubungi kapsida dan dikelilingi membran. RAN terdiri dari 10.794 nukleotida yang mengkodekan 3.419 asam amino.gambar struktur dari virus zika (Rampengan, Novie H, 2016).

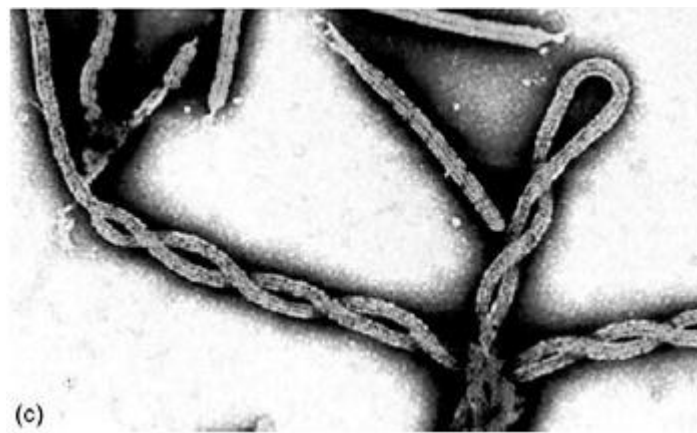


Sumber : *laboratoryinfo.com*

Gambar 2.22 Struktur Virus Zika

10. Virus Ebola

Virus ebola termasuk golongan *Filoviridae*. Virus ini berbentuk huruf U, angka 6, atau lingkaran, tetapi yang paling banyak terlihat di mikrosko elektron ialah struktur tubular panjang. Virus ebola mengandung 1 molekul linear single stranded dengan *negative-sense* RNA yang hampir mirip dengan *Paramyxoviridae*. Gambar adalah virus ebola dari elektron mikroskop yang termasuk anggota keluarga *Filoviridae*.



Sumber : *Brian W. J. Mahy & Marc H. V. Van Regenmortel, 2010:500*

Gambar 2.23 Virus Ebola

Perpindahan virus ebola masuk melalui kontak langsung dari darah, sekret tubuh, organ atau cairan tubuh lainnya dari individu yang terinfeksi. Perpindahan

juga dapat terjadi dari hewan ke manusia melalui kontak dengan jaringan dan cairan tubuh hewan yang terinfeksi.

Gejala yang timbul jika terkena infeksi virus ini dapat dibagi dalam 4 fase: yaitu:

1. Fase A : *Influenza like syndrome*. Terjadi gejala atau tanda nonspesifik seperti panas tinggi, sakit kepala, artalgia, mialgia, nyeri tenggorokan, lemah badan, dan malaise.
2. Fase B: bersifat akut (hari ke 1-6). Terjadi demam persisten yang tidak merespon terhadap obat anti malaria atau antibiotik, sakit kepala, lemah badan yang terus menerus, dan diikuti oleh diare, nyeri perut, anoreksia, dan muntah.
3. Fase C: Pseudo-remisi (hari ke 7-8). selama periode ini penderita merasa sehat dengan konsumsi makanan yang baik. Sebagian penderita dapat sembuh dalam periode ini dan selamat dari penyakit.
4. Fase D: Terjadi agregasi (hari ke 9). Pada beberapa kasus terjadi penurunan kondisi kesehatan yang drastis diikuti oleh gangguan respirasi; dapat terjadi gangguan hemostasis berupa perdarahan pada kulit (petekia) serta gangguan neuropsikiatrik seperti delirium, koma, gangguan kardiovaskular, dan syok hipovolemik.

2.6 Android

2.6.1 Pengertian Android

“Android merupakan sebuah perangkat lunak untuk perangkat *mobile* yang meliputi sistem operasi, *middleware* dan aplikasi inti yang dirilis oleh Google” (Mulyadi, 2010).

F. Gozali (2012) Android adalah perangkat lunak mobile yang mencakup sistem operasi, *middleware* dan aplikasi utama mobile yang bersifat terbuka dengan memisahkan hardware dengan software yang berjalan diatasnya. Device yang berbeda dapat menjalankan suatu aplikasi yang sama dan membangun ekosistem yang lebih kaya untuk para pengembang dan konsumen .

Satyaputra dan Aritonang (2014) Android merupakan sistem operasi yang bersifat open source (sumber terbuka) karena source code dapat dilihat, didownload dan dimodifikasi secara bebas sehingga dapat memberikan kontribusi terhadap pengembangan sistem operasi maupun aplikasi .

2.6.2 Sejarah android

Awalnya, Google Inc membeli Android Inc yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android dibentuklah Open Hanset Alliance, konsorsium (pembiayaan bersama suatu proyek atau perusahaan yang dilakukan oleh dua atau lebih lembaga) dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, Hie, Intel, Motorola, Qualcomm, T-Mobile,dan Nvidia (Nazrudin Safaat H, 2015:1).

Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Hansdset Alliance menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Dilain pihak, Google merilis kode-kode android dibawah Lisensi Apache, sebuah lisensi perangkat lunak dan *open platform* perangkat selular. Didunia ini terdapat dua jenis *disributor* sistem android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail Services (GMS) dan kedua adalah yang benar-

benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution*(OHD)(Nazrudin Safaat H, 2015:1).

Sekitar september 2007 Google mengenalkan *smartphone* Nexus One diproduksi oleh HTC Cooperation dan tersedia di pasaran pada 5 Januari 2010 yang menggunakan android sebagai sistem operasinya. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android AEM Holdings, Atheros Communication, diproduksi oleh Asustek. OHA mengumumkan produk perdana mereka, android perangkat *mobile* yang merupakan modifikasi kernel Linux. Sejak Android dirilis telah dilakukan berbagai pembaharuan berupa perbaikan *bug* dan penambahan fitur baru (Nazrudin Safaat H, 2015:2).

Pada masa saat ini kebanyakan vendor-vendor *smartphone* sudah memproduksi *smartphone* berbasis android, vendor-vendor itu antara lain HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus, Sciphone, Wayteq, Sony Ericson, Acer, Philips, T-Mobile, Nexian, IMO, Asus dan masih banyak lagi vendor *smartphone* di dunia yang memproduksi android. Hal ini karena android adalah sistem operasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh vendor manapun (Nazrudin Safaat H, 2015:3).

Selain sistem operasi di *smartphone*, android juga dipakai pada sistem operasi Tablet PC. Faktor pesatnya pertumbuhan android adalah *platform* sangat lengkap baik itu sistem operasinya, aplikasi dan *Tool Development*, Market aplikasi android serta dukungan yang sangat tinggi dari komunitas *Open Source* di dunia, sehingga android terus berkembang pesat dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia (Nazrudin Safaat H, 2015:3).

2.6.3 Android SDK

Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mengembangkan atau menciptakan 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 buat oleh Google.

Beberapa fitur-fitur android yang paling penting adalah (Nazrudin Safaat H, 2015:5) :

1. *Framework* : apliasi yang mendukung pengganti komponen dan *reuseable*
2. Dalvik *Virtual Machine* dioptimalkan untuk perangkat *mobile*.
3. *Integrated Browser* berdasarkan *engine open source* WebKit.
4. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D
5. Berdasarkan spesifikasi opengl ES 1.0 (Opsional Ekselerasi *Hardware*)
6. SQLite untuk penyimpanan data.
7. *Media support* yang mendukung *audio*, *video*, dan gambar.
8. MP3, AAC, AMR, 3G, dan WiFi (tergantung *hardware*)
9. Kamera, GPS, Kompas, dan *Accelerometer* (tergantung *hardware*)
10. Lingkungan *Development* yang lengkap dan termasuk perangkat emulator, *tools*.

2.6.4 Versi Android

nama tiap versi android diambil dari sebuah makanan penutup di sebuah restoran dan di urut sesuai dengan abjad. Berikut adalah versi-versi android (Masruri, M Hilmi, 2015: 4):

1. Android Versi Beta

Android versi beta ini dirilis pada tanggal 5 November 2007, dimana Android pertama kali muncul dan masih belum digunakan oleh orang seperti saat ini.

2. Android Versi 1.0

Android versi 1.0 ini dirilis pada tanggal 23 November 2008, ponsel pertama yang menggunakan sistem operasi ini yaitu HTC GI(*Dream*). Pada android 1.0 ada yang memberi pernyataan dengan nama Angel Gake dan ada juga yang memberi pernyataan dengan nama Appel Pie. Fitur-fitur yang diberikan Android 1.0 adalah sebagai berikut:

- a. Sudah terintegrasi dengan Gmail
- b. Tersedia layanan Google *talk*
- c. Terdapat *media player* dan masih banyak lagi fitur yang lainnya.

3. Android versi 1.1

Android versi 1.1 ini dirilis pada tanggal 9 Maret 2009. Pada versi ini telah dilakukan perbaikan-perbaikan terhadap sejumlah permasalahan yang ditemukan pada android versi 1.0 sebelumnya. Begitu juga pada android versi 1.1 ini juga ada perbedaan nama versi tersebut yaitu : “*Battenberg*” dan “*Banana Bread*”.

4. Android Versi 1.5

Android versi 1.5 ini dirilis oleh Google pada tanggal 30 April 2009. Pada versi inilah Google pertama kali memberikan nama kode atau *name code* untuk versi android, yaitu dengan nama “*Cupcake*” yang artinya “Kue Mangkuk”.

5. Android versi 1.6

Android versi 1.6 ini dikeluarkan oleh Google pada tanggal 15 September 2009. Kali ini Google memberikan kode nama “*Donut*”. Yaitu sebuah nama makanan yang sudah tidak asing lagi ditelinga kita dan pada versi ini pula telah hadir android *market* baru. Android 1.6 dirilis untuk memperbaiki kesalahan *reboot* pada sistem operasi serta perubahan fitur pada antarmuka kamera dan integrasi pencarian yang lebih baik.

6. Android versi 2.0 dan 2.1

Android versi 2.0 ini dirilis pada tanggal 26 Oktober 2009 selang waktu 3 bulan Google telah merilis kembali sistem operasi android dengan versi 2.1 tepatnya pada tanggal 12 Januari 2011, kedua versi tersebut diberi nama kode yang sama, yaitu “*Eclair*”, dimana *Eclair* adalah makanan penutup yang biasanya dihidangkan bersama dengan secangkir kopi hangat pada sebuah restoran.

7. Android versi 4.0

Android versi 4.0 dirilis pada tanggal 19 Oktober 2011 yang diberikan nama “*Ice Cream Sandwich*”. Versi ini merupakan keluaran terbaru dari Google yang membawa “*Honeycomb*” pada *smartphone*.

8. Android versi 4.1, 4.2, dan 4.3

Dikenalkan pada 27 Juni 2012 pada konferensi Google I/O. Perkembangan banyak terjadi pada versi ini yang membuatnya menjadi android yang

tercepat dan terhalus yang pernah ada. Pengalaman berbeda bisa kita dapati dari *interface* serta pengenalan *Google Search* dengan kemampuan baru.

“Jelly Bean” dianggap sebagai versi android yang memiliki kinerja paling cepat dan terhalus. Resmi diperkenalkan disebuah konferensi Google I/O pada Juni 2012, versi android ini mampu memberikan pengalaman luar biasa bagi penggunaanya dengan menyajikan tampilan *interface* serta *Google Search* yang dibekali kemampuan baru.

Android Versi 4.4

Sebelum resmi dirilis, para pengamat *gadget* memprediksi bahwa untuk versi lanjutan dari Jelly Bean akan diberi nama “*Key Lime Pie*”, tapi ternyata android versi ini diberi kode nama “Kitkat”.

9. Android Versi 5.0

Android “Lollipop” ini dirilis pada tanggal 15 Oktober 2014. Meskipun pada saat itu Lollipop baru dalam masa percobaan akan tetapi komentar yang masuk terbilang bagus.

2.7.5 Arsitektur Android

Secara garis besar arsitektur android dapat dijelaskan dan digambarkan sebagai berikut (Nazrudin Safaat, H:2015):

1. *Applications dan Widgets*

Applications dan *widget* 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.

2. *Application frameworks*

Android adalah “*open development platform*” yaitu android menawarkan kepada pengembang atau memberi kepada pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *service background*, mengatur *alarm*, dan menambah status *notification*, dan sebagainya.

Komponen-komponen yang termasuk dalam applications framework adalah sebagai berikut:

- a. *Views*
- b. *Content Provider*
- c. *Resource Manager*
- d. *Notification Manager*
- e. *Activity manager*

3. *Libraries*

Libraries ini adalah layer dimana fitur-fitur android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan diatas kernel, layer ini meliputi berbagai library C/C++ inti seperti libc dan SSL, serta:

- a. *Libraries* media untuk pemutaran media *audio* dan *video*
- b. *Libraries* untuk manajemen tampilan
- c. *Librariesgraphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
- d. *Libraries Slite* untuk dukungan database.

- e. *Libraries* SSL dan webkit terintegrasi dengan web browser dan *security*
- f. *Librarieslivesoftware* mencakup *modern* web browser dengan *engine embedded web view*
- g. *Libraries* 3D mencakup implementasi openGL ES 1.0 API'S

4. 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:

2.7 *Augmented Reality*

Gorbala dan Hariadi (2010) dalam penelitian Rifa'i, Muhammad., Listiyorini, Tri., & Latubessy, Anastasya (2014) *Augmented Reality* sebagai suatu lingkungan yang memasukkan objek virtual 3D ke dalam lingkungan nyata secara interaktif dalam waktu nyata (*real time*) .

Menurut Edward Jhon (2015) dalam Penelitian Ardianto, Eka ., Hadi Kurniawati., & Winarno, Edy. 2012 *Augmented Reality* sebagai penggabungan benda-benda nyata (*real*) dan maya dilingkungan nyata, berjalan secara interaktif dalam waktu nyata, dan terdapat integrasi hubungan antar benda dalam tiga dimensi, yaitu benda maya yang terintegrasi dalam dunia nyata .

Menurut Saputro, Rujianto Eko & Saputra, Dhanar Intan Surya. 2014 penelitian tentang pengembangan media pembelajaran mengenal organ pencernaan manusia menggunakan teknologi augmented reality mengemukakan bahwa *Augmented Reality* adalah aplikasi mobile yang mengabungkan dan melengkapi built-in komponen dalam ponsel android yang diterapkan di atas realitas fisik menggunakan

kamera untuk mengintegrasikan gambar nyata, *video* atau skenario dalam diri mereka

.2.7.1 Markerless

Rizki (2012), salah satu metode *Augmented Reality* adalah menggunakan metode *markerless augmented reality*, dengan metode ini pengguna tidak perlu menggunakan sebuah marker (penanda) untuk menampilkan elemen-elemen digital sehingga lebih efisien, praktis, menarik dan digunakan kapan pun dan dimanapun .

Menurut Rahman, Abdur., Ernawati., & Coestera, Funny Farady. 2014 penelitian tentang rancang bangun aplikasi informasi universitas bengkulu sebagai panduan pengenalan kampus menggunakan metode *markerless augmented reality* berbasis android mengemukakan bahwa Dalam *markerless*, marker yang dikenali berbentuk posisi perangkat, arah, maupun lokasi. Total Immersion dan Qualcomm adalah salah satu perusahaan yang mengembangkan *Augmented Reality* dengan berbagai macam bentuk teknik *Markerless Tracking* diantaranya sebagai berikut :

1. *Face Tracking* : dengan menggunakan algoritma yang mereka kembangkan, komputer akan mengenali wajah manusia dengan cara mengenali posisi mata, hidung, dan mulut manusia dengan mengabaikan objek-objek disekitarnya.
2. *3D Object Tracking* : teknik 3D objek yang dapat mengenali semua bentuk benda yang ada disekitarnya, seperti mobil, meja, televisi dan lain-lain.
3. *Motion Tracking* : pada teknik ini komputer dapat menangkap gerakan secara real time.
4. *GPS Based Tracking* : teknik ini diarahkan pada smarphone, karena teknologi GPS dan kompas yang tertanam pada smartphone tersebut. Dengan implementasi

augmented reality fitur GPS dapat dimanfaatkan sebagai penentu lokasi yang ingin dituju.

2.8 Pemodelan Tiga Dimensi

Nalwan (1997) dalam penelitian Setiawan, Arif., Tambunan, Toufan Diansyah., & Hendriyanto, Robbi, (2016) tentang Android Augmented Reality Untuk Menampilkan Katalog Furniture Secara Tiga Dimensi (3D) Berdasarkan Objek Marker mengemukakan bahwa pemodelan adalah membentuk suatu benda-benda atau objek dengan cara membuat dan mendesain objek tersebut sehingga terlihat seperti hidup. Ada beberapa aspek yang harus diperhatikan bila membangun model objek yang akan dibuat, kesemuanya memberi kontribusi pada kualitas asil akhir. Hal-hal ini tersebut meliputi metode untuk mendapatkan atau membuat data yang mendeskripsikan objek, tujuan dari model, tingkat kerumitan, perhitungan biaya, kesesuaian dan kenyamanan, serta kemudahan manipulasi model. Proses pemodelan 3D membutuhkan perancangan yang dibagi dengan beberapa tahapan untuk pembentukannya. Seperti objek apa yang ingin dibentuk sebagai objek dasar, metode pemodelan objek 3D, pencahayaan dan animasi gerakan objek sesuai dengan urutan proses yang akan dilakukan.

2.9 Blender

Hendi Hendratman (2015) Blender adalah perangkat lunak sumber terbuka (open source) grafis komputer yang digunakan untuk membuat film animasi, efek visual, mencetak 3D interaktif dan permainan video dengan beberapa fitur termasuk pemodelan 3D, penteksturan, penyunting gambar bitmap, digital sculpting serta fitur lainnya yang dikembangkan oleh neogeo .

2.10 Unity 3D

Rickman Roedvan (2014) Unity merupakan perangkat lunak pembuat game dalam berbagai macam platform seperti *unity Web Windows, Mac, Android, IOS, Xbox, Playstation 3* dan *wii*. Perangkat lunak Unity 3D bisa didapatkan secara gratis, tetapi ada beberapa fitur yang hanya dapat digunakan jika membayar untuk lisensi berbayarnya .

Sudyatmika, dkk. (2014), Unity3D adalah tools yang terintegrasi untuk membuat bentuk objek 3 dimensi seperti visualisasi arsitektur atau animasi 3D real-time. Lingkungan dari pengembangan unity 3D dapat berjalan pada *unity Web multiplatform*. Unity 3D juga dapat membuat game browser yang dijalankan pada ac dan Windows.

Sari (2013), Unity dapat membuat objek dan diberikan fungsi untuk menjalankan objek tersebut. Berikut bagian-bagian dalam Unity :

1. Asset, adalah tempat penyimpanan dalam unity yang menyimpan suara, gambar, video, dan tekstur.
2. Scenes, adalah area yang berisikan konten-konten dalam game, seperti membuat sebuah level, membuat menu, tampilan tunggu, dan sebagainya.
3. Game Objects, adalah barang yang ada di dalam assets yang dipindah kedalam scenes, yang dapat digerakkan, diatur ukurannya dan diatur rotasinya
4. Components, reaksi baru bagi objek seperti collision, memunculkan partikel dan sebagainya
5. Script, yang dapat digunakan dalam unity ada tiga, yaitu Javascript, C# dan BOO.