# BAB II
# LANDASAN TEORI

## 2.1 REKAYASA PERANGKAT LUNAK[15]

Rekayasa perangkat lunak adalah disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan. Secara umum, perekayasa perangkat lunak memakai pendekatan yang sistematis dan terorganisir terhadap pekerjaan mereka karena cara ini seringkali paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Namun demikian, rekayasa ini sebenarnya mencakup masalah pemilihan metode yang paling sesuai untuk satu set keadaan dan pendekataan yang lebih kreatif, informal terhadap pengembangan yang mungkin efektif pada beberapa keadaan. Pengembangan informal sangat cocok untuk pengembangan sistem *e-commerce* berbasis *web* yang membutuhkan gabungan keahlian perangkat lunak dan perancangan grafis.

Proses perangkat lunak adalah serangkaian kegiatan dan hasil-hasil relevannya yang menghasilkan perangkat lunak. Kegiatan-kegiatan ini sebagian besar dilakukan oleh perekayasa perangkat lunak. Kegiatan ini bisa mencakup pengembangan perangkat lunak mulai dari awal, walaupun kenyataannya makin sering terjadi bahwa perangkat lunak yang baru dikembangkan dengan memperluas dan memodifikasi sistem yang telah ada.

Model proses perangkat lunak merupakan representasi abstrak dari proses perangkat lunak. Setiap model proses mempresentasikan suatu proses dari sudut pandang tertentu sehingga hanya memberikan informasi parsial mengenai proses tersebut. Berikut ini merupakan beberapa model proses yang sangat umum (kadangkala disebut paradigma proses) dan mengetengahkan model-model ini dari sudut pandang arsitektural. Jadi, kita melihat kerangka kerja proses tapi bukan perincian dari suatu kegiatan tertentu.

### 2.1.1 Model 'Air Terjun'

Model pertama yang diterbitkan untuk proses pengembangan perangkat lunak diambil dari proses rekayasa lain (Royce, 1970). Berkat penurunan dari satu fase ke fase lainnya, model ini dikenal sebagai 'model air terjun' atau siklus hidup perangkat lunak. Tahap-tahap utama dari model ini memetakan kegiatan-kegiatan pengembangan dasar yaitu :

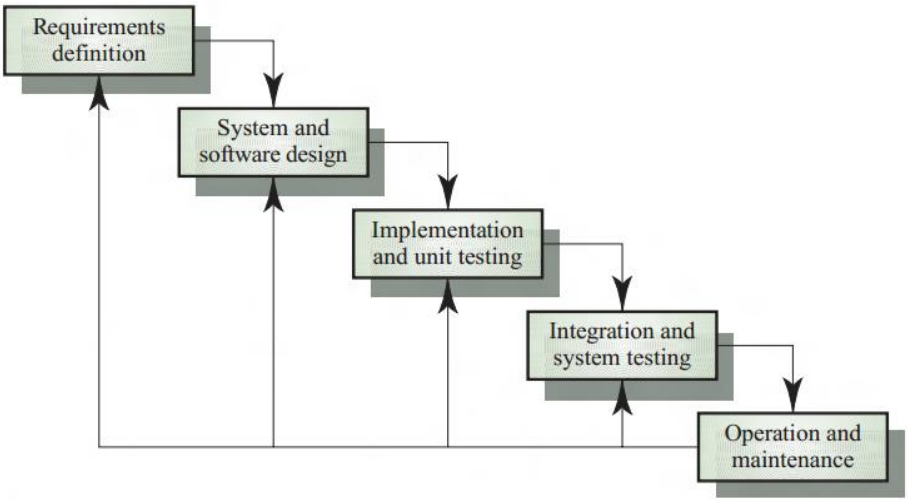
a. Analisis dan definisi persyaratan. Pelayanan, batasan, dan tujuan sistem ditentukan melalui konsultan dengan *user* sistem. Persyaratan ini kemudian didefinisikan secara rinci dan berfungsi sebagai spesifikasi sistem.

b. Perancangan sistem dan perangkat lunak. Proses perancangan sistem membagi persyaratan dalam sistem perangkat keras atau perangkat lunak. Kegiatan ini menentukan arsitektur sistem secara keseluruhan. Perancangan perangkat lunak melibatkan identifikasi dan deskripsi abstraksi sistem perangkat lunak yang mendasar dan hubungan-hubungannya.

c. Implementasi dan pengujian unit. Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian unit melibatkan verifikasi bahwa setiap unit telah memenuhi spesifikasinya.

d. Integrasi dan pengujian sistem. Unit program atau program individual diintegrasikan dan diuji sebagai sistem yang lengkap untuk menjamin bahwa persyaratan sistem telah dipenuhi. Setelah pengujian sistem, perangkat lunak dikirim kepada pelanggan.

e. Operasi dan pemeliharaan. Biasanya (walaupun tidak seharusnya), ini merupakan fase siklus hidup yang paling lama. Sistem diinstal dan dipakai. Pemeliharaan mencakup koreksi dari berbagai *error* yang tidak ditemukan pada tahap-tahap terdahulu, perbaikan atas implementasi unit sistem dan pengembangan pelayanan sistem, sementara persyaratan-persyaratan baru ditambahkan.




Gambar 2.1 Model Air Terjun (*Waterfall*)[3]

Pada prinsipnya, hasil dari setiap fase merupakan satu atau lebih dokumen yang disetujui (‘ditanda-tangani’). Fase yang berikutnya tidak boleh dimulai sebelum fase sebelumnya selesai. Pada prakteknya, tahap-tahap ini bertumpang tindih dan memberi informasi satu sama lain. Saat perancangan, masalah dengan persyaratan diidentifikasi, pada saat pengkodean ditemukan masalah perancangan dan seterusnya. Proses perangkat lunak bukanlah model linear sederhana, tetapi melibatkan serangkaian iterasi kegiatan pengembangan. Masalah dengan model air terjun adalah terjadinya pembagian proyek menjadi tahap-tahap yang tidak fleksibel. Komitmen harus dilakukan pada tahap awal proses, dan akan sulit bagi perekayasa untuk menanggapi perubahan persyaratan pelanggan. Dengan demikian, model air terjun harus digunakan hanya ketika persyaratan dipahami dengan baik. Bagaimanapun juga, model air terjun mereflesikan praktek rekayasa. Secara konsekuen, proses perangkat lunak yang berdasarkan pada pendekatan ini masih digunakan untuk pengembangan perangkat lunak, terutama jika merupakan bagian dari sistem proyek rekayasa yang lebih besar.

### 2.1.2 Pengembangan Evolusioner

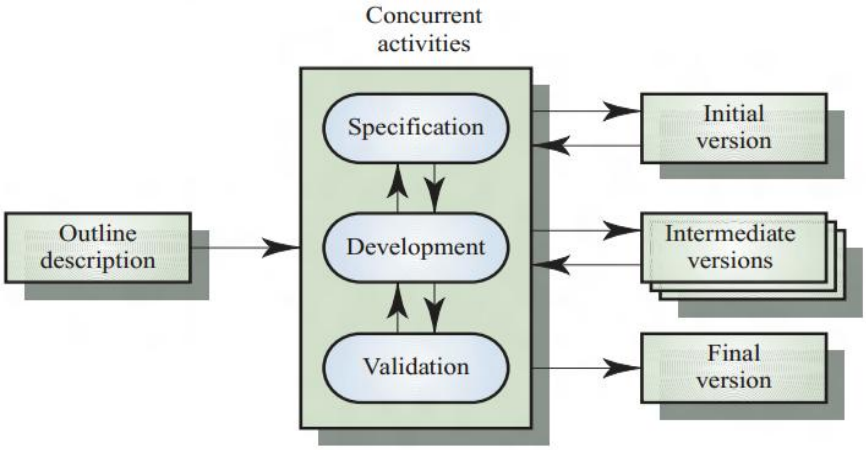
Pengembangan evolusioner berdasarkan pada ide untuk mengembangkan implementasi awal, memperlihatkannya kepada *user* untuk dikomentari, dan memperbaikinya versi demi versi sampai sistem memenuhi persyaratan diperoleh. Tidak ada kegiatan spesifikasi, pengembangan, dan validasi yang

terpisah, alih-alih kegiatan-kegiatan ini dilakukan pada saat yang bersamaan dengan umpan balik yang cepat untuk masing-masing kegiatan.

Ada dua jenis pengembangan evolusioner :

a. Pengembangan eksploratori. Tujuan ini adalah bekerja dengan pelanggan untuk menyelidiki persyaratan mereka dan mengirimkan sistem akhir. Pengembangan dimulai dengan bagian-bagian sistem yang dipahami. Sistem berubah dengan adanya tambahan fitur-fitur baru sesuai usulan pelanggan.

b. Prototipe yang dapat dibuang (*throw-away*). Tujuan pengambangan evolusioner adalah untuk memahami persyaratan pelanggan dan dengan demikian mengembangkan definisi persyaratan yang lebih baik untuk sistem. Prototipe berkonsentrasi pada eksperimen, dengan persyaratan pelanggan yang tidak dipahami dengan baik.




Gambar 2.2 Pengembangan Evolusioner[3]

Pendekatan evolusioner terhadap pengembangan perangkat lunak seringkali lebih efektif dari pendekatan air terjun dalam menghasilkan sistem yang memenuhi kebutuhan langsung dari pelanggan. Keuntungan proses perangkat lunak yang berdasarkan pada pendekatan evolusioner adalah bahwa spesifikasi dapat dikembangkan secara inkremental. Sementara *user* mendapat pemahaman yang lebih baik dari masalah mereka, sistem perangkat lunak dapat mereflesikannya.

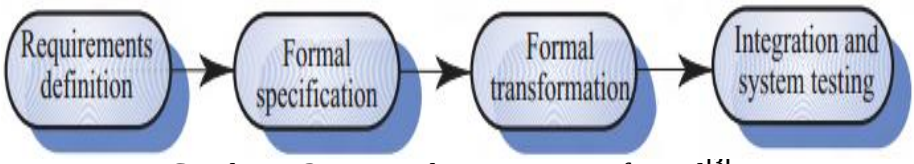
### 2.1.3 Pengembangan Sistem Formal

Pengembangan sistem formal merupakan pendekatan terhadap pengembangan perangkat lunak yang memiliki kesamaan dengan model air

terjun, tetapi proses pengembangannya didasarkan pada transformasi matematis dari spesifikasi sistem menjadi program yang dapat dijalankan. Perbedaan kritis antara pendekatan ini dan model air terjun adalah :

a. Spesifikasi persyaratan perangkat lunak diperbaiki menjadi spesifikasi formal yang rinci yang dinyatakan dalam notasi matematis.
b. Proses pengembangan perancangan, implementasi, dan pengujian unit digantikan oleh proses pengembangan transformasional dimana spesifikasi formal diperbaiki, melalui serangkaian transformasi menjadi program.

Pada proses transformasi, representasi matematis formal dari sistem secara sistematis diubah menjadi representasi sistem yang lebih rinci, tetapi tetap benar secara matematis. Setiap langkah menambahkan perincian sampai spesifikasi formal diubah menjadi program yang ekivalen. Transformasikan sampai cukup dekat sehingga usaha verifikasi transformasi tidak berlebihan. Dengan demikian dapat dijamin, dengan menganggap tidak adanya *error* verifikasi, bahwa program merupakan implementasi yang benar dari spesifikasi.



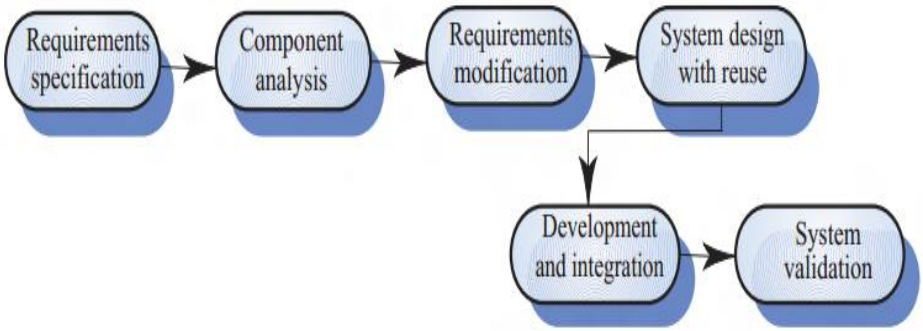

Gambar 2.3 Pengembangan sistem formal[13]

Keuntungan pendekatan transformasional dibandingkan dengan penyetujuan bahwa suatu program telah memenuhi spesifikasinya adalah bahwa jarak antara setiap transformasi lebih kecil daripada jarak antara spesifikasi dan program. Bukti program sangat panjang dan tidak praktis untuk sistem berskala besar. Pendekatan transformasional yang terdiri dari serangkaian langkah yang lebih kecil akan lebih mudah ditelusuri. Namun demikian, pemilihan transformasi apa yang dipakai merupakan pekerjaan yang membutuhkan keahlian, dan membuktikan hubungan transformasi adalah sesuatu yang sulit.

### 2.1.4 Pengembangan Berorientasi Pemakaian Ulang

Pemakaian ulang terjadi pada sebagian besar proyek perangkat lunak. Hal ini biasanya terjadi secara informal ketika orang yang bekerja di proyek tersebut mengetahui adanya rancangan atau kode yang mirip dengan yang dibutuhkan. Mereka mencari rancangan atas kode ini, memodifikasinya sebagaimana dibutuhkan, dan menggabungkannya dalam sistem. Pemakaian ulang informal ini terjadi dengan mengabaikan proses generik yang dipakai. Namun demikian, dalam beberapa tahun belakangan ini, suatu pendekatan terhadap pengembangan perangkat lunak (rekayasa perangkat lunak berdasarkan komponen) yang berdasarkan atas pemakaian ulang telah muncul dan penggunaannya bertambah luas.

Pendekatan yang berorientasi pemakaian ulang ini bergantung pada sejumlah besar komponen perangkat lunak yang dapat diulang, yang bisa didapat, dan beberapa kerangka kerja integrasi untuk komponen-komponen ini. Kadangkala, komponen-komponen ini merupakan sistem juga (COTS atau *Commercial Off-The-Shell systems* / Sistem Siap Beli Komersial) yang dapat digunakan untuk memberikan fungsionalitas khusus seperti format teks, perhitungan numerik, dan lain-lain.




Gambar 2.4 Pengembangan berorientasi pemakaian ulang[13]

Sementara tahap spesifikasi persyaratan dan tahap validasi dapat dibandingkan dengan proses lain, tahap pertengahan pada proses berorientasi pemakaian ulang ini ternyata berbeda. Tahap-tahap ini adalah :

a. Analisis komponen. Jika diketahui spesifikasi persyaratan, komponen-komponen untuk implementasi spesifikasi tersebut akan dicari. Biasanya, tidak ada kesesuaian yang tepat dan komponen yang dapat dipakai hanya memberikan sebagian dari fungsionalitas yang dibutuhkan.

b. Modifikasi persyaratan. Pada tahap ini, persyaratan dianalisis dengan menggunakan informasi mengenai komponen yang telah didapat. Persyaratan kemudian dimodifikasi untuk mereflesikan komponen yang tersedia. Jika modifikasi tidak mungkin dilakukan, maka kegiatan analisis komponen bisa diulang untuk mencari solusi alternatif.

c. Perancangan sistem dengan pemakaian ulang. Pada fase ini, kerangka kerja sistem dirancang, atau kerangka kerja yang telah ada dipakai ulang. Perancang memperhitungkan komponen yang dipakai ulang dan mengatur kerangka kerja untuk menyesuaikan. Beberapa perangkat lunak yang baru mungkin perlu dirancang jika komponen yang dapat dipakai ulang tidak tersedia.

d. Pengembangan dan integrasi. Perangkat lunak yang tidak dapat dibeli akan dikembangkan dan komponen dan sistem COTS diintegrasikan untuk membentuk sistem. Integrasi sistem, pada model ini, dapat merupakan bagian dari proses pengembangan dan bukan merupakan kegiatan yang terpisah.

Model yang berorientasi pemakaian ulang mempunyai keuntungan nyata yaitu mengurangi besarnya perangkat lunak yang akan dikembangkan, serta memperkecil biaya dan resiko. Keuntungan ini biasanya memungkinkan penyelesaian perangkat lunak dengan cepat. Akan tetapi, kompromi persyaratan sangatlah penting dan bisa menghasilkan sistem yang tidak memenuhi kebutuhan sebenarnya dari *user*. Labih jauh lagi, kontrol terhadap evolusi sistem akan hilang sementara versi baru dari komponen yang dapat dipakai ulang tidak terkontrol oleh organisasi yang memakai komponen ini.

## 2.2 PENGUJIAN PERANGKAT LUNAK[15]

Tahap pengujian komponen berhubungan dengan pengujian berfungsinya komponen yang teridentifikasi dengan jelas. Komponen ini dapat berupa fungsi atau sekumpulan metode yang digabungkan menjadi modul atau objek. Saat pengujian integrasi, komponen diintegrasikan untuk membentuk subsistem atau sistem yang lengkap. Pengujian harus terfokus pada interaksi antara komponen dan fungsionalitas dan kinerja sistem sebagai satu kesatuan. Bagaimanapun, tak akan terelakkan lagi bahwa cacat komponen yang terlewat pada pengujian awal akan ditemukan pada saat pengujian integrasi.

Ketika menguji sistem kritis, spesifikasi rinci setiap komponen perangkat lunak digunakan oleh tim independen untuk membuat uj bagi sistem tersebut. Namun demikian, pada sebagian besar kasus lain, pengujian merupakan proses yang lebih intuitif karena tidak ada waktu untuk menulis spesifikasi rinci mengenai bagian sistem perangkat lunak. Di lain pihak, *interface* komponen sistem utama dispesifikasi dan *programmer* individu dan tim *programmer* mengambil tanggung jawab untuk perancangan, pengembangan, dan pengujian komponen-komponen ini. Pengujian biasanya didasarkan pada pemahaman intuitif mengenai bagaimana komponen ini harus beroperasi.

### 2.2.1 Pengujian Cacat

Tujuan pengujian cacat (*defect testing*) adalah mengungkap cacat laten pada sistem perangkat lunak sebelum sistem diserahkan. Ini berlawanan dengan pengujian validasi yang ditujukan untuk mendemonstrasikan bahwa sistem telah memenuhi spesifikasinya. Pengujian validasi menuntut sistem berlaku dengan benar dengan menggunakan kasus uji penerimaan. Uji cacat yang berhasil merupakan uji yang menyebabkan sistem berlaku tidak benar dan dengan demikian mengungkap adanya cacat. Hal ini menekankan fakta penting mengenai pengujian. Pengujian ini menunjukkan keberadaan, bukan tidak adanya kesalahan program.

Kasus uji merupakan spesifikasi *input* untuk menguji dan *output* yang diharapkan dari sistem ditambah pernyataan apa yang sedang diuji. Data uji merupakan *input* yang telah disesuaikan untuk menguji sistem. Data uji

kadang-kadang dibangkitkan secara otomatis. Pembangkitan kasus uji otomatis, tidak mungkin dilakukan karena *output* uji tidak dapat diramalkan.

a. Pengujian Kotak Hitam

Pengujian fungsional atau pengujian kotak hitam (*black-box testing*) merupakan pendekatan pengujian yang ujinya diturunkan dari spesifikasi program atau komponen. Sistem merupakan 'kotak hitam' yang perilakunya hanya dapat ditentukan dengan mempelajari *input* dan *output* yang berkaitan. Nama lain untuk cara ini adalah *pengujian fungsional* karena penguji hanya berkepentingan dengna fungsionlitas dan bukan implementasi perangkat lunak.

b. Paritisi Ekuivalensi

Data *input* untuk program biasanya masuk dalam sejumlah kelas yang berbeda. Kelas-kelas ini memiliki karakteristik sama, misalnya bilangan positif, bilangan negatif, *string* tanpa *blank*, dan sebagainya. Program biasanya berlaku dengan cara yang dapat dibangdingkan dengan semua anggota kelas. Karena perilaku ekuivalen ini, kelas-kelas ini kadang-kadang disebut partisi atau domain ekuivalensi. Satu pendekatan sistematis bagi pengujian cacat didasarkan atas identifikasi semua partisi ekuivalensi yang harus ditangani oleh program. Kasus uji dirancang sehingga *input* atau *output* berada pada partisi ini.

c. Pengujian Struktural

Pengujian struktural (*structural testing*) merupakan pendekatan terhadap pengujian yang diturunkan dari pengetahuan struktur dan implementasi perangkat lunak. Pendekatan ini kadang-kadang disebut pengujian 'kotak putih' ('*white-box*' *testing*), pengujian 'kotak kaca', atau 'pengujian kotak jernih' untuk membedakannya dari pengujian kotak hitam. Pengujian struktural biasanya diterapkan untuk unit program yang relatif kecil seperti subrutin atau operasi terkait dengan suatu objek. Sebagaimana ditunjukkan oleh namanya, penguji dapat menganalisis kode dan menggunakan pengetahuan mengenai struktur komponen untuk menurunkan data uji. Analisis kode dapat digunakan untuk menemukan berapa kasus uji yang dibutuhkan untuk menjamin bahwa semua

*statement* pada program atau komponen dieksekusi paling tidak satu kali pada proses pengujian.

d. Pengujian Jalur

Pengujian jalur (*path testing*) adalah strategi pengujian struktural yang bertujuan untuk melatih setiap jalur eksekusi independen melalui komponen atau program. Jika setiap jalur independen dieksekusi, maka semua *statement* pada komponen harus dieksekusi paling tidak satu kali. Labih jauh lagi, semua *statement* kondisional diuji untuk kasus *true* dan *false*. Pada proses pengembangan berorientasi objek, pengujian jalur dapat digunakan ketika menguji metode yang terkait dengan objek.

### 2.2.2 Pengujian Integrasi

Begitu komponen-komponen program individual telah teruji, mereka harus diintegrasi untuk membentuk sistem parsial atau lengkap. Proses integrasi ini melibatkan pengembangan sistem dan pengujian sistem yang dihasilkan untuk mengetahui adanya masalah yang muncul dari interaksi komponen. Pengujian integrasi harus dikembangkan dari spesifikasi sistem dan pengujian integrasi harus dimulai segera setelah versi beberapa komponen sistem tersedia.

Kesulitan utama yang muncul pada pengujian integrasi adalah lokalisasi eror yang ditemukan pada saat proses tersebut. Terdapat interaksi yang rumit antara komponen sistem dan ketika ditemukan *output* yang menyimpang, mungkin sulit untuk menemukan sumber eror tersebut. Cara memudahkan lokaslisai eror, harus selalu menggunakan pendekatan ikremental terhadap integrasi dan pengujian sistem. Pada awalnya, Anda harus mengintegrasi konfigurasi sistem minimal dan menguji sistem ini. Anda kemudian menambahkan komponen pada konfigurasi minimal ini dan mengujinya setelah setiap inkremen ditambahkan.

a. Pengujian *Top-down* dan *Bottom-up*

Strategi uji *top-down* dan *bottom-up* menunjukkan pendekatan yang berbeda terhadap integrasi sistem. Komponen sistem tingkat tinggi diintegrasikan dan diuji sebelum perancangan dan implementasinya selesai pada integrasi *top-down*. Sedangkan pada integrasi *bottom-up*,

komponen tingkat rendah diintegrasikan dan diuji sebelum komponen tingkat yang lebih tinggi dikembangkan.

Pengujian *top-down* merupakan bagian integral dari proses pengembangan *top-down* dengan proses pengembangan dimulai dengan komponen tingkat tinggi dan berjalan ke bawah menelusuri hierarki komponen. Program dinyatakan sebagai satu komponen abstrak dengan sub-komponen direpresentasikan dengan *stub* (potongan). *Stub* memiliki *interface* yang sama dengan komponen tetapi fungsionalitasnya sangat terbatas. Setelah komponen tingkat paling atas telah diprogram dan diuji, sub-komponennya diimplementasi dan diuji dengan cara yang sama. Proses ini berlanjut sampai komponen tingkat paling bawah diimplementasikan. Seluruh sistem dengan demikian telah diuji.

Sebaliknya, pengujian *bottom-up* melibatkan integrasi dan pengujian modul pada tingkat-tingkat yang lebih rendah pada hierarki, dan kemudian menelusuri hierarki modul ke atas sampai modul terakhir diuji. Pendekatan ini tidak menuntut kelengkapan rancangan arsitektural sistem sehingga pendekatan tersebut dapat mulai pada tahap awal proses pengembangan. Cara ini dapat digunakan jika sistem memakai ulang dan mengubah komponen dari sistem lain.

b. Pengujian *Interface*

Pengujian *interface* dilakukan ketika modul atau subsistem diintegrasi untuk membuat sistem yang lebih besar. Setiap modul atau subsistem memiliki *interface* yang terdefinisi yang dipanggil oleh komponen program lain. Tujuan pengujian *interface* adalah mendeteksi kesalahan yang mungkin telah masuk ke dalam sistem karena eror *interface* atau asumsi *invalid* mengenai *interface*.

Bentuk pengujian ini terutama penting untuk pengembangan berorientasi objek, terutam ketika objek dan kelas objek dipakai ulang. Objek terutama didefinisikan oleh *interface*-nya dan dapat diapakai ulang dalam kombinasi dengan objek yang berbeda pada sistem yang berbeda. Eror *interface* tidak dapat dideteksi dengan pengujian objek individu.

Eror merupakan hasil dari interaksi antara objek dan bukan perilaku terisolasi dari satu objek.

c. Pengujian Stres

Begitu sistem telah terintegrasi sepenuhnya, adalah mungkin untuk menguji sistem akan adanya properti baru yang muncul seperti kinerja dan keandalan. Uji kinerja harus dirancang untuk menjamin bahwa sistem dapat memproses beban yang diberikan. Ini biasanya melibatkan perencanaan serangkaian uji dengan beban ditambah secara tetap hingga kinerja sistem menjadi tidak dapat diterima.

Beberapa kelas sistem dirancang untuk menangani beban yang spesifik. Sebagai contoh, sistem pengolahan transaksi dapat dirancang untuk memproses sampai 100 transaksi per detik; sistem operasi dapat dirancang untuk menangani sampai 200 terminal yang terpisah. Pengujian stres melanjutkan uji-uji ini melewati beban rancangan maksimum sistem sampai sistem gagal. Jenis pengujian ini memiliki dua fungsi:

- Pengujian stres menguji perilaku kegagalan sistem. Mungkin timbul suatu keadaan melalui kombinasi *event* yang tidak diharapkan ketika beban yang diberikan pada sistem melebihi beban maksimum yang diantisipasi. Penting dalam keadaan ini bahwa kegagalan sistem tidak menyebabkan kerusakan data atau kerugian yang tidak diharapkan dari layanan *user*. Pengujian stres memeriksa apakah memberi beban yang lebih kepada sistem menyebabkannya ‘kegagalan lunak’ dan bukan runtuh dengan bebannya.
- Pengujian stres memberi tekanan pada sistem dan dapat menyebabkan munculnya cacat yang biasanya tidak mewujudkan diri mereka. Walaupun dapat diperdebatkan bahwa cacat ini tidak mungkin mengakibatkan kegagalan sistem dalam pemakaian normal, mungkin ada kombinasi yang tidak biasa dari keadaan normal yang direplikasi oleh pengujian stres.

Pengujian stres terutama relevan bagi sistem terdistribusi yang berdasarkan jaringan prosesor. Sistem-sistem ini seringkali menunjukkan degradasi yang parah jika diberi beban berat. Jaringan menjadi terbebani dengan data koordinasi yang harus dipertukarkan oleh berbagai proses sehingga proses menjadi lebih lambat dan lebih lambat sementara mereka menunggu data dari proses lain.

### 2.2.3 Pengujian Berorientasi Objek

Saya menyarankan adanya dua aktivitas dasar pada proses pengujian. Dua aktivitas ini adalah pengujian komponen dengan sistem diuji secara individu dan pengujian integrasi dengan sekumpulan komponen diintegrasi menjadi subsistem dan sistem akhir untuk pengujian. Aktivitas-aktivitas ini sama-sama dapat diterapkan pada sistem berorientasi objek. Bagaimanapun, ada perbedaan penting antara sistem berorientasi objek dan sistem yang dikembangkan dengan menggunakan model fungsional :

- Objek sebagai komponen individu seringkali lebih besar dari fungsi tunggal.
- Objek yang diintegrasi menjadi subsistem biasanya *loosely coupled* dan tidak ada '*top*' yang jelas bagi sistem.
- Jika objek dipakai ulang, penguji mungkin tidak memiliki akses ke kode sumber komponen untuk analisis.

Perbedaan-perbedaan ini berarti bahwa pendekatan pengujian kotak putih yang berdasar analisis kode harus diperluas untuk mencakup objek yang lebih besar dan bahwa pendekatan alternatif terhadap pengujian integrasi harus dipakai. Namun demikian, begitu sistem telah diintegrasi, kenyataan bahwa sistem tersebut telah dikembangkan sebagai sistem berorientasi objek harus jelas bagi *user* sistem.

a. Pengujian Kelas Objek

Pendekatan terhadap liputan uji berhubungan dengan pemberian jaminan bahwa semua *statement* program dieksekusi paling tidak sekali dan bahwa semua jalur yang melalui program dieksekusi. Ketika menguji objek, liputan uji yang lengkap harus mencakup :

- Pengujian dengan isolasi dari semua operasi yang berhubungan dengan objek tersebut;
- *Setting* dan interogasi semua atribut yang berhubungan dengan objek;
- Pelatihan objek dengan semua status yang mungkin. Ini berarti bahwa semua *event* yang menyebabkan perubahan status pada objek harus disimulasikan.

b. Integrasi Objek

Ketika sistem berorientasi objek dikembangkan, tingkat integrasi lebih tidak jelas. Jelas, operasi dan data diintegrasi untuk membentuk objek dan kelas objek. Pengujian kelas objek ini berhubungan dengan pengujian unit. Tidak ada ekuivalensi langsung dengan pengujian modul pada sistem berorientasi objek. Akan tetapi, Murphy *et al.* (1994) mengusulkan bahwa kelompok kelas yang bekerja dalam kombinasi untuk menyediakan satu set layanan harus diuji bersama. Mereka menamakan ini pengujian kelompok (*cluster*).

### 2.2.4 *Workbench* Pengujian

Pengujian merupakan fase proses perangkat lunak yang mahal dan memerlukan kerja keras. Oleh karenanya, alat bantu pengujian merupakan salah satu alat bantu perangkat lunak pertama yang dikembangkan. Alat-alat bantu ini kini memberikan serangkaian fasilitas dan penggunaannya secara signifikan memperkecil biaya proses pengujian. Alat bantu pengujian yang berbeda dapat diintegrasikan ke dalam *workbench* (meja kerja) pengujian.

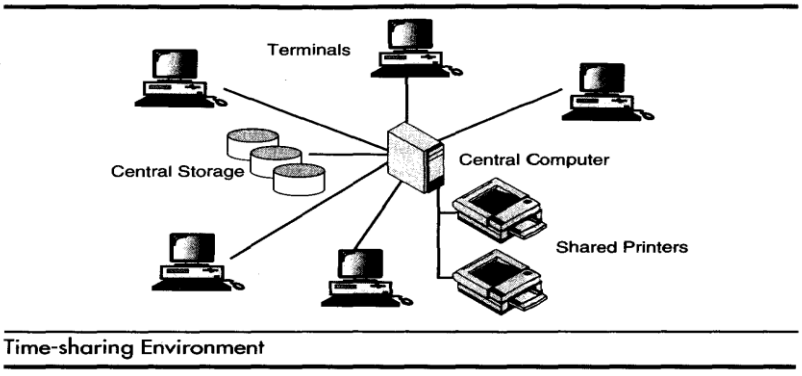
Usaha dan waktu yang signifikan biasanya diperlukan untuk membuat *workbench* pengujian komprehensif. *Workbench* uji yang lengkap dengan demikian hanya digunakan ketika sistem besar sedang dikembangkan. Sistem-sistem seperti ini, biaya pengujian menyeluruh bisa sampai 50 persen biaya pengembangan total sehingga efektif dalam hal biaya jika dilakukan investasi pada penunjang alat bantu *case* kualitas tinggi untuk pengujian.

## 2.3 JARINGAN KOMPUTER

Jaringan komputer merupakan gabungan antara teknologi komputer dan teknologi telekomunikasi. Gabungan teknologi ini melahirkan pegolahan data yang dapat didistribusikan, mencakup pemakaian *database*, *software* aplikasi dan peralatan *hardware* secara bersamaan, sehingga penggunaan komputer yang sebelumnya hanya berdiri sendiri, kini telah diganti dengan sekumpulan komputer yang terpisah-pisah akan tetapi saling berhubungan dalam melaksanakan tugasnya, sistem seperti inilah yang disebut jaringan komputer.[16]

### 2.3.1 Sejarah Jaringan Komputer[12]

Kebutuhan akan komunikasi untuk komputer pada tahun 40-an dan 50-an hanyalah bersifat dasar dan minimal. Prosesor berkomunikasi dengan periferalnya melalui peralatan *input*/*output* (I/O) pada jarak yang pendek dan bekerja pada kecepatan yang sangat rendah. Tahun 1960-an lahir konsep *timesharing*, dimana pengguna dihubungkan ke komputer melalui suatu *dumb terminal*.




Gambar 2.5 *Time-shared computer system*[3]

Tahun 1970-an, teknologi IC (*integrated circuit*) dan mikroprosesor mulai berkembang sehingga memungkinkan munculnya komputer pribadi yang dapat dipasang di rumah-rumah. Perkembangan teknolog ini secara drastis mengubah cara pandang orang tentang komputer. Munculnya teknologi jaringan lokal (*Local Area Network*) dalam tahun 1980-an melengkapi komputer dengan kemampuan berkomunikasi dengan komputer lainnya. Kondisi ini menyebabkan terjadinya migrasi dari konsep pemrosesan secara tersentralisasi (*centralized computing*) menjadi konsep pemrosesan secara terdistribusi (*distribution computing*).

Sistem atau prosesor utam diletakkan secara terpusat dalam konsep *centralized computing*. Semua komputer yang letaknya berjauhan dihubungkan menggunakan *link* secara langsung ke prosesor utama tersebut. Semua informasi (*database*) terletak di sistem pusat. Prosesor atau komputer utama didistribusikan pada lokasi-lokasi yang berbeda pada kasus *distributed computing*. Masing-masing komputer tersebut mempunyai sebagian atau seluruh duplikat dari data yang ada di komputer pusat. Pengguna mengakses informasi dari prosesor yang terdekat yang secara periodik memutakhirkan data yang ada di dalam *database*-nya. Trend pada masa datang adalah migrasi menuju lingkungan *distributed computing* karena perkembangan prosesor dan koneksi *switching* serta intelegensia dapat terletak pada level CPE.

### 2.3.2 Jenis Jaringan Komputer[14]

Berdasarkan area, jaringan komputer dibedakan menjadi lima :

a. *Personal Area Network* (PAN)

PAN merupakan jaringan komputer yang dibentuk oleh beberapa buah komputer dengan peralatan nonkomputer (seperti : *printer*, mesin *fax*, telepon seluler, PDA, dan *handphone*). Sebuah PAN dapat dibangun menggunakan teknologi *wire* dan *wireless network*. Teknologi *wire* PAN biasanya memanfaatkan perangkat USB dan *FireWire*. Sedangkan *wireless* PAN menggunakan teknologi *Bluetooth*, WiFi, dan *Infrared*. Cakupan area sebuah PAN sangat terbatas, yaitu sekitar 6 hingga 9 meter. Namun, perkembangan teknologi telah membuat sebuah PAN dapat menjangkau area yang lebih luas.

b. *Local Area Network* (LAN)

*Local Area Network* (LAN) adalah jaringan komputer yang dibangun pada area yang terbatas, seperti ruangan, rumah, kantor, gedung, kampus. Sebuah LAN dapat terdiri atas puluhan hingga ratusan buah komputer. LAN mendukung kecepatan transfer data cukup tinggi.

c. *Metropolitan Area Network* (MAN)

*Metropolitan Area Network* (MAN) merupakan jaringan komputer yang meliputi area sebuah kota. Teknologi yang digunakan oleh MAN mirip dengan LAN, hanya saja areanya lebih besar dan komputer yang

dapat dihubungkan dengan jaringan pun lebih banyak dibandingkan LAN. MAN dapat berupa sebuah gabungan jaringan komputer beberapa buah sekolah atau beberapa buah kampus. MAN telah diimplementasikan menggunakan teknologi *wire* maupun *wireless network*. *Wireless* MAN dapat menjangkau area yang sulit dijangkau oleh kabel. Salah satu implementasi *wireless network* MAN adalah WiMAX.

MAN dapat memanfaatkan jaringan televisi kabel jenis *coaxial* dan serat optik. Negara-negara yang sudah maju telah memanfaatkan teknologi serat optik pada jaringan TV kabel, sehingga dapat mengangkut data berukuran *gigabit* dengan sangat cepat. Pelanggan TV kabel dapat menikmati akses *internet* berkecepatan tinggi sambil menonton acara TV kesukaannya.

d. *Wide Area Network* (WAN)

*Wide Area Network* (WAN) merupakan jaringan komputer yang meliputi area geografis sangat besar, seperti antarkota, antarnegara, antarbenua. WAN dapat menghubungkan LAN atau MAN yang dipisahkan oleh jarak yang sangat jauh. Cara menghubungkan kedua jarak yang berjauhan biasanya digunakan saluran telepon atau salauran komunikasi publik (umum). Contoh implementasi WAN adalah Internet.

e. Internet

Internet dapat dikategorikan sebagai WAN (*Wide Area Network*) yang bersifat khusus. Ada beberapa hal yang membedakan Intenet dengan WAN, salah satunya yaitu protokol yang digunakan. Intenet menggunakan protokol khusus yang disebut TCP/IP. Ciri lainnya yang membedakan Internet dengan WAN yaitu :

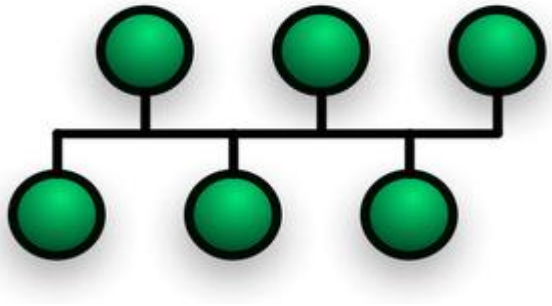
- Servis Internet atau layanan khas Internet
- Model pengoperasian
- Pengalamatan yang unik
- Sistem penamaan *domain* (*domain name system*)
- Sistem penentuan rute tujuan (*routing*)

## 2.4 TOPOLOGI JARINGAN KOMPUTER[14]

Topologi adalah suatu cara menghubungkan komputer yang satu dengan komputer lainnya sehingga membentuk jaringan. Cara yang saat ini banyak digunakan adalah bus, token-ring, star dan peer-to-peer network. Masing-masing topologi ini mempunyai ciri khas, dengan kelebihan dan kekurangannya sendiri.

### 2.4.1 Topologi Bus

Topologi Bus menggunakan sebuah kabel *backbone* dan semua *host* terhubung secara langsung pada kabel tersebut.



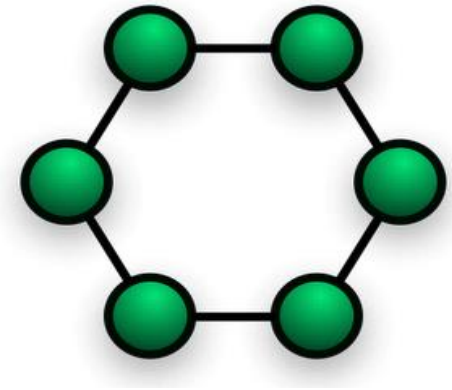
Gambar 2.6 Topologi Bus[8]

Tabel 2.1 Keuntungan dan kerugian Topologi Bus

| Keuntungan | Kerugian |
|---|---|
| - Proses instalasi mudah<br>- Biaya instalasi murah<br>- Penambahan *node* dapat dilakukan dengan mudah<br>- Bekerja baik pada *network* skala kecil | - Merupakan teknologi lama yang sudah *out of date*.<br>- Jika kabel putus atau rusak maka *network* lumpuh total<br>- Proses *troubleshooting* cukup sukar<br>- Manajemen pada *network* skala besar tidak dapat dilakukan |

### 2.4.2 Topologi *Ring*

Topologi *Ring* menghubungkan *host* dengan *host* lainnya membentuk lingkaran tertutup atau *loop*.



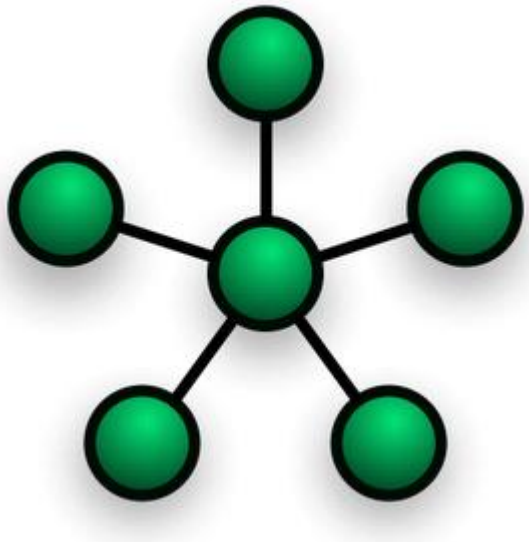
Gambar 2.7 Topologi *Ring*[8]

Tabel 2.2 Keuntungan dan kerugian Topologi *Ring*

| Keuntungan | Kerugian |
|---|---|
| - Proses instalasi mudah<br>- Biaya instalasi murah<br>- Penambahan *node* dapat dilakukan dengan mudah<br>- Bekerja baik pada *network* skala kecil | - Peka kesalahan<br>- Pengembangan jaringan lebih kaku |

### 2.4.3 Topologi *Star*

Topologi *Star* menghubungkan semua komputer pada sentral atau konsentrator. Biasanya konsentrator berupa perangkat *hub* atau *switch*.



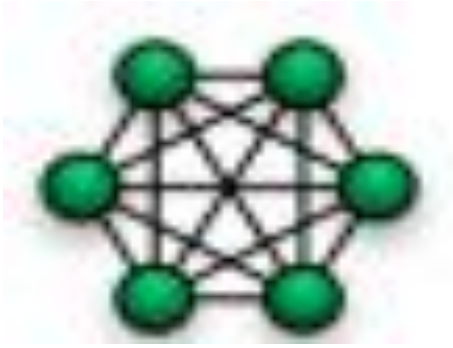
Gambar 2.8 Topologi *Star*[8]

Tabel 2.3 Keuntungan dan kerugian Topologi Star

| Keuntungan | Kerugian |
|---|---|
| - Proses instalasi mudah<br>- Biaya instalasi murah<br>- Penambahan *node* dapat dilakukan dengan mudah<br>- Proses *troubleshooting* mudah<br>- Jika salah satu kabel putus atau rusak maka *network* masih dapat berfungsi<br>- Manajemen *network* terpusat dan memudahkan untuk *network* skala besar | - Biaya instalasi cukup mahal<br>- Jika *hub* atau *switch* rusak maka *network* akan lumpuh total |

### 2.4.4 Topologi *Mesh* atau *Fully-Mesh*

Topologi *Mesh* menghubungkan setiap komputer secara *point-to-point*. Artinya semua komputer akan saling terhubung satu-satu sehingga tidak dijumpai ada *link* terputus. Topologi ini biasanya digunakan pada lokasi yang kritis, seperti instalasi nuklir.



Gambar 2.9 Topologi *Mesh* atau *Fully-Mesh*[8]

Tabel 2.4 Keuntungan dan kerugian Topologi *Mesh* atau *Fully-Mesh*

| Keuntungan | Kerugian |
|---|---|
| - Sangat *fault tolerant*. Karena banyak *link* dengan setiap *node*. | - Biaya instalasi cukup mahal<br>- Proses instalasi sukar<br>- Proses manajemen sukar<br>- Proses *troubleshooting* sukar |

Topologi *Mesh* merupakan jenis topologi yang digunakan oleh Internet. Setiap *link* menghubungkan suatu *router* dengan *router* lainnya.

## 2.5 MODEL REFERENSI OSI[14]

OSI (*Open System Interconnection*) *Reference Model* atau model referensi OSI adalah sebuah model untuk jaringan komputer yang dikembangkan oleh *International Organization for Standardization* (ISO) di Eropa pada tahun 1977. Model OSI ini disebut juga model OSI tujuh lapis atau OSI *seven layer* model yang mulai diperkenalkan pada tahun 1984. Model OSI dapat digunakan untuk menjelaskan cara kerja jaringan komputer secara logika. Dahulu kala, komunikasi data yang melibatkan komputer-komputer dari *vendor* yang berbeda sangat sulit dilakukan. Masing-masing *vendor* menggunakan protokol dan format data yang berbeda-beda, sehingga ISO membuat suatu arsitektur komunikasi yang dikenal sebagai model OSI yang mendefinisikan standar untuk menghubungkan komputer-komputer dari *vendor* yang berbeda.

Pada awalnya model OSI dimaksudkan untuk keperluan standarisasi protokol jaringan komputer. Namun, ide tersebut gagal diwujudkan. Ada beberapa faktor yang menyebabkan kegagalan, yaitu :

- Model OSI dianggap terlalu kompleks. Beberapa fungsi (seperti metode komunikasi *connectionless*) dianggap kurang memadai. Sementara fungsi lainnya (seperti *flow control* dan *error connection*) diulang-ulang di beberapa *layer*.
- OSI menggunakan 7 buab *layer* yang dianggap terlalu rumit dan hanya mempertimbangkan aspek “politik” dibandingkan “teknik”. *Layer Presentation* dan *Session* dianggap tidak berguna (kosong), sedangkan *layer Data Link* dan *Network* sangat penuh dengan fitur-fitur.
- Adanya campur tangan politik menyebabkan OSI dianggap sebagai “makhluk” buatan Kementrian Telekomunikasi Eropa dan pemerintah Amerika Serikat. Campur tangan birokrasi untuk mengatur protokol jaringan komputer ternyata menimbulkan sentimen negatif dari para pengguna jaringan komputer,

sehingga implementasi jaringan komputer yang menggunakn model OSI jarang dijumpai di luar benua Eropa.

- Pertumbuhan Internet berbasis protokol TCP/IP yang sangat pesat telah membuat model OSI menjadi kurang populer dan kurang diminati. Internet dapat berkembang tanpa model OSI.

Hingga saat ini, model OSI hanya merupakan "model ideal" dan digunakan sebagai acuan untuk memudahkan mempelajari bagaimana protokol-protokol jaringan berfungsi dan berinteraksi. Berikut adalah fungsi dan penjelasan masing-masing *layer* yang ada di model OSI :

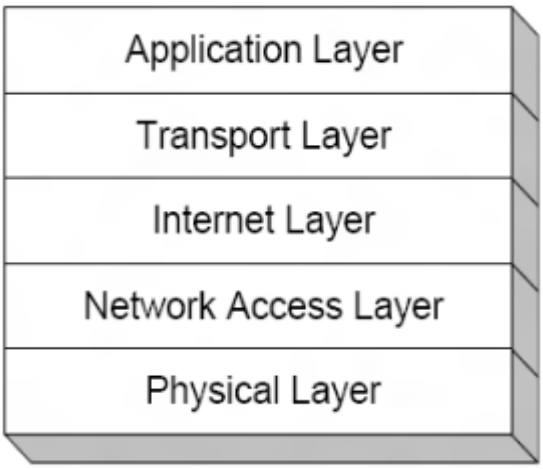
- *Layer* 7 (*Application*), berfungsi sebagai antarmuka (penghubung) aplikasi dengan fungsionalitas jaringan, mengatur bagaimana aplikasi dapat mengakses jaringan, dan kemudian membuat pesan-pesan kesalahan. Pada *layer* inilah sesungguhnya *user* "berinteraksi dengan jaringan". Contoh protokol yang berada pada lapisan ini : FTP, telnet, SMTP, HTTP, POP3, NFS.
- *Layer* 6 (*Presentation*), berfungsi untuk mentranslasikan data yang hendak ditransmisikan oleh aplikasi ke dalam format yang dapat ditransmisikan melalui jaringan. Protokol yang berada pada level ini adalah sejenis *redirector software*, seperti *network shell* (semacam *Virtual Network Computing*) atau *Remote Desktop Protocol* (RDP). Kompresi data dan enkripsi juga ditangani oleh *layer* ini.
- *Layer* 5 (*Session*), berfungsi untuk mendefinisikan bagaimana koneksi dimulai, dipelihara, dan diakhiri. Selain itu, di level ini juga dilakukan resolusi nama. Beberapa protokol pada *layer* ini : NETBIOS (protokol yang dikembangkan IBM, menyediakan layanan untuk *layer presentation* dan *layer application*), NETBIOS *Extended User Interface* (atau disebut NETBEUI, merupakan protokol pengembangan dari NETBIOS digunakan pada Microsoft *Networking*), ADSP (*Apple Talk Data Stream Protocol*), PAP (*Printer Access Protocol*, protokol untuk printer Postscript pada jaringan *Apple Talk*).
- *Layer* 4 (*Transport*), berfungsi untuk memecah data menjadi paket-paket data serta memberikan nomor urut setiap paket sehingga dapat disusun kembali setelah diterima. Paket yang diterima dengan sukses akan diberi tanda (*acknowledgement*). Sedangkan paket yang rusak atau hilang di tengah jalan

akan dikirim ulang. Contoh protokol yang digunakan pada *layer* ini seperti : UDP, TCP, SPX.

- *Layer* 3 (*Network*), berfungsi untuk mendefinisikan alamat-alamat IP, membuat *header* untuk paket-paket, dan melakukan *routing* melalui *internetworking* dengan menggunakan *router* dan *switch layer* 3. Pada *layer* ini juga dilakukan proses deteksi *error* dan transmisi ulang paket-paket yang *error*. Contoh protokol yang digunakan seperti : IP, IPX.
- *Layer* 2 (*Data Link*), berfungsi untuk menentukan bagaimana bit-bit data dikelompokkan menjadi format yang disebut *frame*. Pada level ini terjadi *error correction, flow control,* pengalamatan perangkat keras (MAC *Address*), dan menentukan bagaimana perangkat-perangkat jaringan seperti *bridge* dan *switch layer* 2 beroperasi. Menurut spesifikasi IEEE 802, *layer* ini dikelompokkan menjadi dua, yaitu *Logical Link Control* (LLC) dan *Media Access Control* (MAC). Contoh protokol yang digunakan pada layer ini adalah : *Ethernet* (802.2 & 802.3), *Token Bus* (802.4), *Token Ring* (802.5), *Demand Priority* (802.12).
- *Layer* 1 (*Physical*), berfungsi untuk medefinisikan media transmisi jaringan, metode pensinyalan, sinkronasi bit, arsitektur jaringan (seperti halnya *Ethernet* atau *Token Ring*), topologi jaringan, dan pengkabelan. Selain itu, level ini juga mendefinisikan bagaimana *Network Interface Card* (NIC) berinteraksi dengan *media wire* atau *wireless*. *Layer physical* berkaitan langsung dengan besaran fisis seperti listrik, magnet, gelombang. Data biner dikodekan berbentuk sinyal yang dapat ditransmisi melalui media jaringan.

## 2.6 ARSITEKTUR DAN PROTOKOL JARINGAN TCP/IP (*TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL*)[11]

Adapun lapisan-lapisan (*layer*) dalam arsitektur jaringan komputer memiliki tugas spesifik serta memiliki protokol tersendiri. ISO (*International Standard Organization*) telah mengeluarkan suatu standar untuk arsitektur jaringan komputer yang dikenal dengan nama *Open System Interconnection* (OSI). Standar ini terdiri dari 7 lapisan protokol yang menjalankan fungsi komunikasi antara 2 komputer. Ada lima lapisan dalam TCP/IP sebagai berikut:




Gambar 2.10 TCP/IP *Layer*[11]

Semua fungsi dari lapisan-lapisan arsitektur OSI telah tercakup oleh arsitektur TCP/IP walaupun jumlahnya berbeda. Adapun rincian fungsi masing-masing *layer* arsitektur TCP/IP adalah sebagai berikut:

1. *Application Layer* merupakan lapisan dalam arsitektur TCP/IP yang berfungsi mendefinisikan aplikasi-aplikasi yang dijalankan pada jaringan. Oleh karena itu, terdapat banyak protokol pada lapisan ini, sesuai dengan banyaknya aplikasi TCP/IP yang dapat dijalankan. Contohnya adalah SMTP (*Simple Mail Transfer Protocol*) untuk pengiriman *e-mail*, FTP (*File Transfer Protocol*) untuk *transfer file*, HTTP (*Hyper Text Transfer Protocol*) untuk aplikasi *web*, NNTP (*Network News Transfer Protocol*) untuk distribusi *news group* dan lain-lain.
2. *Transport Layer* mendefinisikan cara-cara untuk melakukan pengiriman data antara *end to end host* secara handal. Lapisan ini menjamin bahwa informasi

yang diterima pada sisi penerima adalah sama dengan informasi yang dikirimkan pada pengirim.

3. *Internet Layer* mendefinisikan bagaimana hubungan dapat terjadi antara dua pihak yang berada pada jaringan yang berbeda seperti *Network Layer* pada OSI. Lapisan ini juga bertugas untuk menjamin agar suatu paket yang dikirimkan dapat menemukan tujuannya dimanapun berada dalam jaringan *internet* yang terdiri atas puluhan juta *host* dan ratusan ribu jaringan lokal,. Oleh karena itu, lapisan ini memiliki peranan penting terutama dalam mewujudkan *internetworking* yang meliputi wilayah luas (*Worldwide Internet*).
4. *Network Access Layer* mempunyai fungsi yang mirip dengan *Data Link Layer* pada OSI. Lapisan ini mengatur penyaluran data *frame-frame* data pada media fisik yang digunakan secara handal. Lapisan ini biasanya menggunakan servis untuk deteksi dan koreksi kesalahan dari data yang ditransmisikan. Beberapa contoh protokol yang digunakan pada lapisan ini adalah X.25 jaringan publik, *Ethernet* untuk jaringan *Ethernet*, AX.25 untuk jaringan Paket Radio dan sebagainya.
5. *Physical Layer* (lapisan fisik) merupakan lapisan terbawah yang mendefinisikan besaran fisik seperti media komunikasi, tegangan, arus, dan sebagainya. Lapisan ini dapat bervariasi tergantung pada media komunikasi pada jaringan yang bersangkutan. TCP/IP bersifat fleksibel sehingga dapat mengintegrasikan berbagai jaringan dengan media fisik yang berbeda-beda.

## 2.7 *VIRTUAL PRIVATE SERVER* (VPS)[9]

*Virtual Private Server* lebih dikenal dengan singkatan VPS. Ini adalah sebuah metode untuk mempartisi sebuah komputer *server* menjadi beberapa *server* yang sepertinya *server-server* tersebut berdiri sendiri. Ia seolah-olah sebagai *server* mandiri dan berlaku benar-benar seperti layaknya sebuah komputer. Sebuah *server* dibagi menjadi beberapa *account reseller* dalam *reseller hosting*. Para pengguna menggunakan *server* secara bersama-sama yang artinya menggunakan *resources* secara bersama-sama. VPS bertindak seolah-olah menjadi komputer mandiri, pelanggan seolah-olah mempunyai sebuah *server* sendiri. Meskipun kenyataannya

dalam satu komputer terdapat beberapa *account* VPS, namun satu *account* dengan *account* yang lain tidak akan saling mempengaruhi.

## 2.8 UBUNTU *SERVER*[13]

Ubuntu merupakan salah satu distribusi sistem operasi berbasis linux yang dikembangkan dari distribusi Debian. Proyek Ubuntu disponsori oleh Canonical Ltd sebuah perusahaan milik Mark Shuttleworth yang berkedudukan diBelanda. Nama Ubuntu diambil dari nama sebuah konsep ideologi di Afrika Selatan. Kata Ubuntu berasal dari bahasa kuno Afrika, yang berarti rasa perikemanusian terhadap sesama manusia. Tujuan dari distribusi Linux Ubuntu adalah membawa semangat yang terkandung di dalam Ubuntu ke dalam dunia perangkat lunak.

Sistem operasi Ubuntu Server dapat dipasang pada beberapa tipe arsitektur komputer diantaranya Intel X86, AMD 64, ARM, SPARC, PowerPC, Itanium64 bahkan pada Playstation 3. Ubuntu Server memiliki kebutuhan sistem minimum yang harus dipenuhi diantaranya adalah sebagai berikut:

- Prosesor 300Mhz
- *Memory* 64MB
- HDD 500MB
- VGA 640x480

Kebutuhan diatas merupakan minimum ketersediaan pada komputer untuk dapat bekerja dengan Ubuntu Server. Namun demikian, semakin besar layanan maupun penggunaan *server*, sebaiknya disediakan sumber daya komputer yang lebih tinggi untuk memperlancar proses komputasi yang terjadi pada *server*.


Gambar 2.11 Ubuntu dikembangkan berdasarkan distribusi Debian[13]

## 2.9 *BANDWIDTH*[2]

Tabel 2.5 Unit *Bandwidth*

| Unit-unit Lebar Pita | Akronim | Persamaan |
|---|---|---|
| *Bit per second* | bps | 1 bps = ukuran dasar *bandwidth* |
| *Kilobit per second* | Kbps | 1 Kbps = 1.000 bps |
| *Megabit per second* | Mbps | 1 Mbps = 1.000.000 bps |
| *Gigabit per second* | Gbps | 1 Gbps = 1.000.000.000 bps |

*Bandwidth* sangat berpengaruh terhadap kualitas presentasi suatu data *stream*. Kondisi jaringan juga mempengaruhi *bandwidth*, selain itu hal yang perlu diperhatikan adalah ukuran data *stream* harus sesuai dengan kapasitas *bandwidth* jaringan. Kompresi data dan penggunaan *buffer* digunakan mengatasinya.

## 2.10 *STREAMING*[5]

*Streaming* adalah sebuah teknologi untuk memainkan *file video* atau *audio* secara langsung ataupun dengan *pre-recorder* dari sebuah mesin *server* (*web server*). *File video* atau *audio* yang terletak pada sebuah *server* dapat secara langsung dijalankan pada komputer *client* sesaat setelah ada permintaan dari *users* sehingga proses *download file video* atau *audio* yang menghabiskan waktu cukup lama dapat dihindari.

Saat *file video* dan *audio* di-*stream* maka akan terbentuk sebuah *buffer* di komputer *client* dan *data video* atau *audio* tersebut akan mulai di-*download* ke dalam *buffer* yang telah terbentuk pada mesin *client*. *Buffer* akan terisi penuh dalam waktu sepersekian detik dan secara otomatis *file video* atau *audio* akan dijalankan oleh sistem. Sistem akan membaca informasi dari *buffer* sambil tetap melakukan proses *download file* sehingga proses *streaming* tetap berlangsung ke mesin *client*. *Delay* waktu sesaat sebelum *file video* atau *audio* dijalankan berkisar antara 10-30 detik.

## 2.11 *REAL TIME STREAMING PROTOCOL* (RTSP)[4]

RTSP (*Real Time Streaming Protocol*) adalah protokol pada tingkat aplikasi untuk mengontrol penyampaian data secara *real-time*. *Client* dapat mengontrol jalannya presentasi dengan RTSP, misalnya melakukan *rewind, play, fast-forward, pause, resume,* dan *stop* terhadap aliran data. Sumber aliran data dapat meliputi keduanya, *webcast* dan *on-demand*. Protokol RTSP memiliki sintaks dan operasi yang mirip dengan protokol HTTP (*HyperText Transfer Protocol*). Operasi protokol RTSP berdasarkan alokasi dan penggunaan sumber media (*resource*) di *server* memiliki beberapa *state* :

Tabel 2.6 *State* operasi protokol RTSP

| | |
|---|---|
| *SETUP* | *Server* mengalokasikan *resource* dan suatu *session* RTSP dimulai |
| *PLAY* dan *RECORD* | Mulai mentransmisikan data media yang dialokasikan via *SETUP* |
| *PAUSE* | Media dalam keadaan berhenti sementara tanpa membebaskan *resource* di *server* |
| *TEARDOWN* | Membebaskan *resource* yang berhubungan dengan media tersebut, dan *session* RTSP dibuang dari *server* |

## 2.12 *REAL TIME MESSAGING PROTOCOL* (RTMP) [2]

RTMP adalah sebuah protokol yang dikembangkan oleh Adobe System untuk *streaming audio, video* dan data melalui internet, antara Flash Player dan *server*.

RTMP memiliki tiga variasi :

1. Plain protokol berjalan pada tcp menggunakan *port* 1935.
2. RTMPT yang dienkapsulasi dalam http meminta *firewall traverse*.
3. RTMPS bekerja sama dengan RTMPT di dalam HTTPs koneksi.

## 2.13 RED5 *SERVER*[2]

*Red5 Server* adalah sebuah aplikasi *server* yang memungkinkan pengiriman data aktual dari *client* ke *client* yang lainnya. *Red5 Server* menyediakan kemampuan media *streaming* yang *powerfull* dan lingkungan pengembangan yang fleksibel. *Red5 Server* merupakan *open source* Flash RTMP *server* berbasis Java yang mendukung :

1. *Streaming Audio/Video* (FLV *and* MP3).
2. *Recording Client Streams* (FLV *only*).
3. *Shared Objects.*
4. *Live Stream Publishing.*
5. *Remoting* (AMF).

## 2.14 *WEBINAR*

*Webinar* yang biasa ditulis ‘webinar’ merupakan kependekan dari *Web-based Seminar* atau presentasi, kuliah, *workshop* atau seminar yang di lakukan melalui *interface Web*. *Webinar* bisa diterapkan di bidang pendidikan yaitu Sekolah Jarak Jauh. Seorang dosen bisa memberikan pelajaran dari seberang lautan kepada mahasiswanya di kampus menggunakan teknologi komunikasi. Hubungan interaktif dan diskusi bisa di jalankan, sesi tanya jawab tetap terlaksana dan transfer ilmu tetap berjalan. Asal usul *Web Base Seminar* ini dimulai dari *Real Time Text* Chat. Sebagian besar dari pengguna internet di tahun 1980 sampai 1990-an menggunakan *Internet Relay Chat* (IRC) untuk berkomunikasi dengan teman di tempat lain.[3]

## 2.15 BIGBLUEBUTTON

BigBlueButton merupakan sebuah proyek *open source* yang dibangun dari lima belas komponen *open source* untuk menciptakan sebuah sistem *web conferencing* terintegrasi sehingga memungkinkan untuk presentasi jarak jauh dengan *slide*, *audio*, *video*, *chat* dan *desktop sharing* yang berjalan di MAC, UNIX, atau komputer PC. BigBlueButton berfokus untuk membuat sistem *web conferencing* terbaik untuk pendidikan jarak jauh.[10]BigBlueButton merupakan sebuah aplikasi *server* yang dapat dikonfigurasi (*open source*) sendiri yang lebih mengunggulkan fitur-fiturnya dibidang pendidikan jarak jauh.