**BAB II**

**LANDASAN TEORI**

**2.1 Rekayasa Perangkat Lunak**

**2.1.1 Pengertian Perangkat Lunak**

Perangkat lunak *(software)* adalah program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentsi kebutuhan, model desain, dan cara penggunaan (user manual). Sebuah program komputer tanpa terasiosasi dengan dokumentasinya maka belum dapat disebut perangkat lunak *(software).* Sebuah perangkat lunak juga sering disebut dengan sistem perangkat lunak. Sistem berarti kumpulan komponen yang saling terkait dan mempunyai satu tujuan yang dicapai (Rosa A.S, dan M. Salahuddin, 2014).

Sistem perangkat lunak berarti sebuah sistem yang memiliki komponen berupa perangkat lunak yang memiliki hubungan satu sama lain untuk memenuhi kebutuhan pelanggan *(customer).* Pelanggan *(customer)* adalah orang atau organisasi yang memesan atau membeli perangkat lunak *(software)* dari pengembang perangkat lunak atau bisa dianggap bahwa pelanggan *(customer)* adalah orang atau organisasi yang dengan sukarela mengeluarkan uang untuk memesan atau membeli perangkat lunak. *User* atau pemakai perangkat lunak adalah orang yang memiliki kepentingan untuk memakai atau menggunakan pearangkat lunak untuk memudahkan pekerjaan (Rosa A.S, dan M. Salahuddin, 2014).

Menurut O’Brien pada tahun 1999, Perangkat lunak adalah seluruh perintah yang digunakan untuk memproses informasi. Perangkat lunak dapat berupa program atau prosedur. Program adalah kumpulan perintah yang dimengerti oleh komputer sedangkan prosedur adalah perintah yang dibutuhkan oleh pengguna dalam memproses informasi.

Karakter perangkat lunak adalah sebagai berikut :

1. Perangkat lunak dibangun dengan rekayasa *(software engineering)* bukan diproduksi secara manufaktur atau pabrikan.
2. Perangkat lunak tidak pernah usang *(wear out)* karena kecacatan dalam perangkat lunak dapat diperbaiki.
3. Barang produksi pabrikan biasanya komponen barunya akan terus diproduksi, sedangkan perangkat lunak biasanya terus diperbaiki seiring bertambahnya kebutuhan.

**2.1.2 Pengertian Rekayasa Perangkat Lunak**

Rekayasa perangkat lunak adalah sebuah profesi yang dilakukan oleh seorang perekayasa perangkat lunak yang berkaitan dengan pembuatan dan pemeliharaan aplikasi perangkat lunak dengan menerapkan teknologi dan praktik dari ilmu komputer, manajemen proyek, dan bidang-bidang lainya. Teknik rekayasa perangkat lunak akan meningkatkan *fungsionalitas* dan efisiensi aplikasi dan juga kemudahan dan efisiensi dari pengembang perangkat lunak (Simarmata Janner, 2010).

Rekayasa atau teknik adalah penerapan ilmu dan teknologi untuk menyelesaikan permasalahan manusia. Hal ini diselesaikan lewat pengetahuan, matematika, dan pengalaman praktis yang diterapkan untuk mendesain objek atau proses yang berguna. Para praktisi teknik professional disebut (Simarmata Janner, 2010 ).

Rekayasa perangkat lunak *(RPL atau SE [Software Engineering])* adalah satu bidang propesi yang mendalami cara-cara pengembangan perangkat lunak termasuk pembuatan, pemeliharaan, manajemen organisasi pengembangan perangkat lunak, dan sebagainya (Simarmata Janner, 2010 ).

Pada tahun 2004, istilah rekayasa perangkat lunak secara umum digunakan dalam tiga arti, yaitu:

1. Sebagai istilah umum untuk sebagai kegiatan yang dulunya bernama pemograman atau analisis sistem,
2. Sebagai istilah yang luas untuk analisis teknis dari semua aspek-aspek praktis yang bertentangan dengan teori pemograman komputer, dan
3. Sebagai istilah yang luas mewujudkan *advokasi* suatu pendekatan spesifik ke pemograman komputer, satu hal yang mendesak yang diperlakukan sebagai profesi rekayasa dari pada sebuah seni atau kerajianan, dan *advokasi* dari kodifikasi *praktis* yang disarankan dalam bentuk metologi rekayasa perangkat lunak.

Rekayasa perangkat lunak adalah disiplin rekayasa dengan perangkat lunak yang dikembangkan. Biasanya proses melibatkan penemuan pada keinginan klien, menyusunnya di dalam daftar kebutuhan, merancang arsitektur yang mampu mendukung semua kebutuhan, perancangan, pengodean, pengujian, dan pengintegrasian bagian yang terpisah, menguji keseluruhan, penyebaran dan pemeliharaan perangkat lunak. Pemograman hanya menjadi bagian kecil dari rekayasa perangkat lunak (Simarmata Janner, 2010 ).

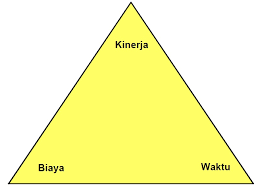
*The Software Engineering Body of Knowledge (SWEBOK)* membagi rekayasa perangkat lunak ke dalam 10 area pengetahuan, yaitu:

1. Kebutuhan perangkat lunak,
2. Perancangan perangkat lunak,
3. Kontruksi perangkat lunak,
4. Pengujian perangkat lunak,
5. Pemeliharaan perangkat lunak,
6. Manajemen konfigurasi perangkat lunak,
7. Manajemen perangkat lunak,
8. Proses perangkat lunak,
9. Metode dan tool perangkat lunak, dan
10. Kualitas perangkat lunak.

Pengertian lain dari Rekayasa perangkat lunak yaitu disiplin ilmu yang lebih fokus pada praktik pengembangan perangkat lunak dan mengirimkan perangkat lunak yang bermanfaat kepada pelanggan *(customer)* atau *user*. Adapun ilmu komputer lebih fokus pada teori dan konsep dasar perangkat komputer. Rekayasa perangkat lunak lebih fokus pada bagaimana membuat perangkat lunak lebih fokus pada bagaimana membuat perangkat lunak yang memenuhi kriteria berikut (Rosa A.S, dan M. Salahuddin, 2014 ).

**2.1.3 Tujuan Rekayasa Perangkat Lunak**

Secara umum tujuan Rekayasa Perangkat Lunak tidak berbeda dengan bidang rekayasa yang lain. Bidang rekayasa akan selalu berusaha menghasilakn output yang kinerjanya tinggi waktu penyelesaian yang tepat dan berbiaya rendah. Secara lebih khusus kita dapat menyetakan tujuan *RPL* adalah sebagai berikut berikut (Rosa A.S, dan M. Salahuddin, 2014 ) :



**Sumber : *(Rosa A.S, dan M. Salahuddin, 2014 )***

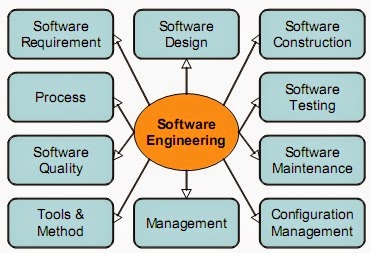
**Gambar 2.1 Tujuan Rekayasa Perangkat Lunak**

Dari Gambar 2.1 dapat diartikan bahwa bidang rekayasa akan selalu berusaha menghasilkan output yang kinerjanya tinggi, biaya rendah dan waktu penyelesaian yang tepat. Secara lebih khusus kita dapat menyatakan tujuan *RPL* adalah berikut (Rosa A.S, dan M. Salahuddin, 2014 ) :

1. Memperoleh biaya produksi perangkat lunak yang rendah.
2. Menghasilkan perangkat lunak yang kinerjanya tinggi,andal dan tepat waktu.
3. Menghasilkan perangkat lunak yang dapat bekerja pada berbagai jenis platform.
4. Menghasilkan perangkat lunak yang biaya perawatannya rendah.

**2.1.4 Ruang Lingkup Rekayasa Perangkat Lunak**

Sesuai definisi yang telah disampaikan sebelumnya, maka ruang lingkup  
RPL dapat digambarkan sebagai berikut.



**Sumber : (*Ratna Wardani, 2012* )**

**Gambar 2.2 Ruang Lingkup Rekayasa Perangkat Lunak**

Penjelasan dari istilah diatas (Ratna Wardani, 2012 ):

1. *Software requirements* berhubungan dengan spesifikasi kebutuhan dan persyaratan perangkat lunak.
2. *Software design* mencakup proses penentuan arsitektur, komponen, antarmuka, dan karakteristik lain dari perangkat lunak.
3. *Software construction* berhubungan dengan detil pengembangan perangkatlunak, termasuk algoritma, pengkodean, pengujian, dan pencarian kesalahan.
4. *Software testing* meliputi pengujian pada keseluruhan perilaku perangkat lunak.
5. *Software maintenance* mencakup upaya-upaya perawatan ketika perangkat lunak telah dioperasikan.
6. *Software configuration* management berhubungan dengan usaha perubahan konfigurasi perangkat lunak untuk memenuhi kebutuhan tertentu.
7. *Software engineering* management berkaitan dengan pengelolaan dan pengukuran *RPL*, termasuk perencanaan proyek perangkat lunak.
8. *Software engineering* *tools* and *methods* mencakup kajian teoritis tentangalat bantu dan metode *RPL*.
9. *Software engineering* process berhubungan dengan definisi, implementasi, pengukuran, pengelolaan, perubahan dan perbaikan proses *RPL*.
10. *Software quality* menitik beratkan pada kualitas dan daur hidup perangkat lunak.

**2.2 SDLC**

**2.2.1 Pengertian SDLC**

SDLC atau *Software Development Life Cycle* atau sering disebut juga dengan *System 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). Seperti halnya proses metamorphosis pada kupu-kupu, untuk menjadi kupu-kupu yang indah maka dibutuhkan beberapa tahap untuk dilalui, sama halnya dengan membuat perangkat lunak, memiliki daur tahapan yang dilalui agar menghasilkan perangkat lunak yang berkualitas (Rosa A.S, dan M. Salahuddin, 2014: 26).

Tahapan-tahapan yang ada pada SDLC secara global adalah sebagai berikut:

1. Inisiasi (*Initiation*)

Tahap ini biasanya ditandai dengan pembuatan proposal proyek perangkat lunak.

1. Pengembangan konsep sistem (*system concept development*) mendefinisikan lingkup konsep termasuk dokumen lingkup sistem, analisis manfaat biaya, manajemen rencana, dan pembelajaran kemudahan sistem.
2. Perencanaan (*Planning*)

Mengembangkan rencana menajemen proyek dan dokumen perencanaan lainya. Menyediakan dasar untuk mendapatkan sumber daya (*resources*) yang dibutuhkan untuk memperoleh solusi.

1. Analisis kebutuhan (*Requirements Analysis*)

Menganalisis kebutuhan pemakai sistem perangkat lunak (*user*) dan mengembangkan kebutuhan *user*. Membuat dokumen kebutuhan fungsional.

1. Desain (*Design*)

Mentransmorfasikan kebutuhan detail menjadi kebutuhan yang sudah lengkap, dokumen desain sistem fokus pada bagaimana dapat memenuhi fungsi-fungsi yang dibutuhkan.

1. Pengembangan (*Development*)

Mengonversi desain ke sistem informasi yang lengkap termasuk bagaimana memperoleh dan melakukan instalasi lingkunagn sistem yang dibutuhkan; membuat basis data dan mempersiapkan prosedur kasus pengujian; mempersiapkan berkas atau *file* pengujian, pengodean, pengompilasian, memperbaiki dan membersihkan program; peninjauan pengujian.

1. Integrasi dan pengujian (*Integration and Test*)

Mendemonstrasikan sistem perangkat lunak bahwa telah memenuhi kebutuhan yang dispesifikasikan pada dokumen kebutuhan fungsional. Dengan diarahkan oleh staf penjamin kualitas (*quality assurance*) dan *user*. Menghasilkan laporan analisis pengujian.

1. Implementasi (*Implementation*)

Termasuk pada persiapan implementasi, implementasi perangkat luak pada lingkungan produksi (lingkungan pada *user*) dan menjalankan resolusi dari permasalahan yang teridentifikasi dari *face* integrasi dan pengujian.

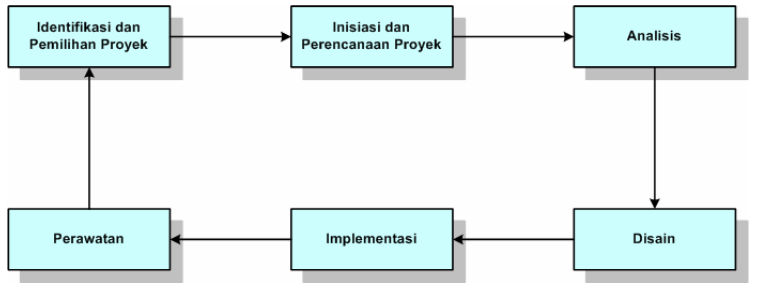
1. Operasi dan pemeliharaan (*Operations and Maintenence*)

Mendeskripsikan pekerjaan untuk mengoperasikan dan memelihara sistem informasi sistem pada lingkungan produksi (lingkungan pada *user*), termasuk implementasi akhir dan masuk pada proses peninjauan

1. Disposisi (*Disposition*)

Mendeskropsikan aktifitas akhir dari pengembangan sistem dan membangun data yang sebenarnya sesuai dengan aktifitas *user.*

berikut adalah gambar *System Development Life Cycle (SDLC)*:



**Sumber : (*Aunur R. Mulyanto, 2008: 18*)**

**Gambar 2.3 *System Development Life Cycle* (SDLC).**

Setiap model yang dikembangkan mempunyai karakteristik sendirisendiri. Namun secara umum ada persamaan dari model-model ini, yaitu (Aunur R. Mulyanto, 2008: 18):

1. Kebutuhan terhadap definisi masalah yang jelas. Input utama dari setiap  
   model pengembangan perangkat lunak adalah pendefinisian masalah yang  
   jelas. Semakin jelas akan semakin baik karena akan memudahkan dalam  
   penyelesaian masalah. bagian penting dari model pengembangan perangkat lunak.
2. Tahapan-tahapan pengembangan yang teratur. Meskipun model-model  
   pengembangan perangkat lunak memiliki pola yang berbeda-beda, biasanya model-model tersebut mengikuti pola umum *analysis–design-coding–testing-maintenance.*
3. *Stakeholder* berperan sangat penting dalam keseluruhan tahapan  
   pengembangan. *Stakeholder* dalam rekayasa perangkat lunak dapat berupa  
   pengguna, pemilik, pengembang, pemrogram dan orang-orang yang terlibat dalam rekayasa perangkat lunak tersebut.
4. Dokumentasi merupakan bagian penting dari pengembangan perangkat  
   lunak. Masing-masing tahapan dalam model biasanya menghasilkan sejumlah tulisan, diagram, gambar atau bentuk-bentuk lain yang harus didokumentasi dan merupakan bagian tak terpisahkan dari perangkat lunak yang dihasilkan.
5. Keluaran dari proses pengembangan perangkat lunak harus bernilai  
   ekonomis. Nilai dari sebuah perangkat lunak sebenarnya agak susah dirupiah-kan. Namun efek dari penggunaan perangkat lunak yang telah  
   dikembangkan haruslah memberi nilai tambah bagi organisasi. Hal ini dapat berupa penurunan biaya operasi, efisiensi penggunaan sumber daya,  
   peningkatan keuntungan organisasi, peningkatan “*image*” organisasi dan  
   lain-lain.

**2.2.2 Model SDLC**

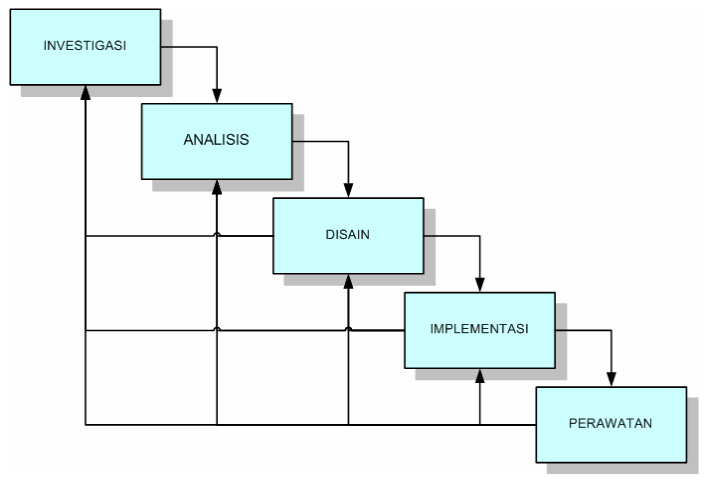
SDLC memiliki beberapa model dalam penerapan tahapan prosesnya.

1. Model *Waterfall*

Model siklus hidup (*life cycle model*) adalah model utama dan dasar dari banyak model. Salah satu model yang cukup dikenal dalam dunia rekayasa perangkat lunak adalah *The Waterfall Model*. Ada 5 tahapan utama dalam *The Waterfall Model*, Disebut *waterfall* (berarti air terjun) karena memang diagram tahapan prosesnya mirip dengan air terjun yang bertingkat (Aunur R. Mulyanto, 2008: 19).

Tahapan-tahapan dalam *The Waterfall Model* secara ringkas adalah  
sebagai berikut (Aunur R. Mulyanto, 2008: 19):

1. Tahap investigasi dilakukan untuk menentukan apakah terjadi suatu masalah atau adakah peluang suatu sistem informasi dikembangkan. Pada tahapan ini studi kelayakan perlu dilakukan untuk menentukan apakah sistem informasi yang akan dikembangkan merupakan solusi yang layak.
2. Tahap analisis bertujuan untuk mencari kebutuhan pengguna dan organisasi serta menganalisa kondisi yang ada (sebelum diterapkan sistem informasi yang baru).
3. Tahap disain bertujuan menentukan spesifikasi detil dari komponen komponen sistem informasi (manusia, *hardware, software, network* dan data) dan produk-produk informasi yang sesuai dengan hasil tahap analisis.
4. Tahap implementasi merupakan tahapan untuk mendapatkan atau  
   mengembangkan *hardware* dan *software* (pengkodean program),  
   melakukan pengujian, pelatihan dan perpindahan ke sistem baru.
5. Tahapan perawatan (*maintenance*) dilakukan ketika sistem informasi sudah dioperasikan. Pada tahapan ini dilakukan monitoring proses, evaluasi dan perubahan (perbaikan) bila diperlukan.



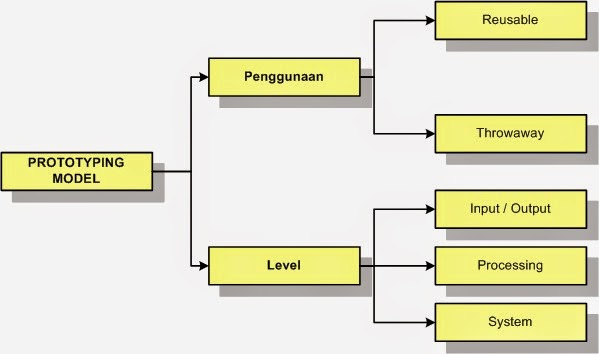
**Sumber : (*Aunur R. Mulyanto, 2008: 20*)**

**Gambar 2.4. *The Waterfall Model***

1. Model *Prototype*

*Prototyping* adalah salah satu pendekatan dalam rekayasa perangkat lunak yang secara langsung mendemonstrasikan bagaimana sebuah perangkat lunak atau komponen-komponen perangkat lunak akan bekerja dalam lingkungannya sebelum tahapan konstruksi aktual dilakukan.

*Prototyping* *model* dapat diklasifikasikan menjadi beberapa tipe seperti terlihat pada gambar 2.5.



**Sumber : (*Aunur R. Mulyanto, 2008: 20*)**

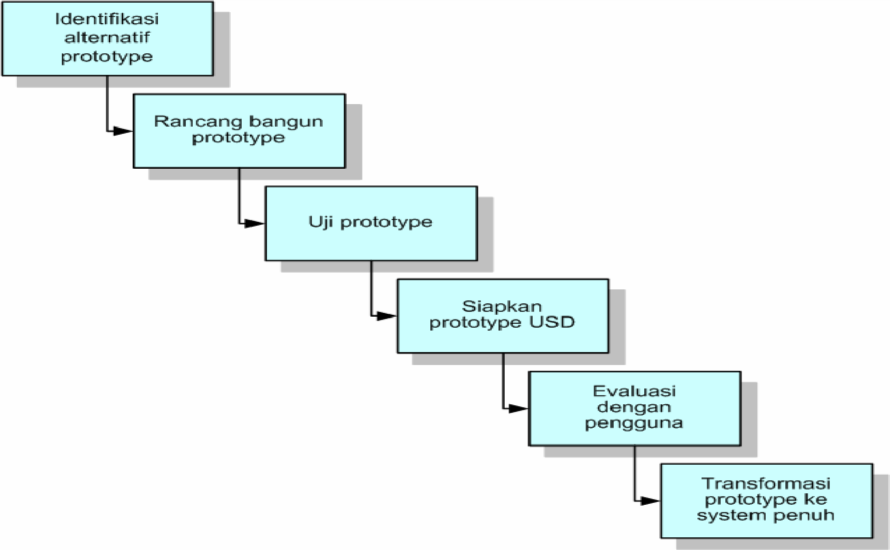
**Gambar 2.5. Klasifikasi *Prototyping Model***

Penjelasan dari istilah diatas (Aunur R. Mulyanto, 2008: 21):

1. *Reusable prototype*: *Prototype* yang akan ditransformasikan menjadi produk *final*.
2. *Throwaway prototype*: *Prototype* yang akan dibuang begitu selesai menjalankan maksudnya.
3. *Input/output prototype*: *Prototype* yang terbatas pada antar muka pengguna (user interface).
4. *Processing prototype*: *Prototype* yang meliputi perawatan file dasar dan proses-proses transaksi.
5. *System prototype*: *Prototype* yang berupa model lengkap dari perangkat lunak.

Strategi utama dalam *prototyping* adalah kerjakan yang mudah  
terlebih dahulu dan sampaikan hasil kepada pengguna sesegera mungkin, membagi *prototyping* dalam enam tahapan (Aunur R. Mulyanto, 2008: 21) yaitu:

1. Identifikasi kandidat *prototyping*. Kandidat dalam kasus ini meliputi user *interface* (menu, dialog, *input* dan *output*), file-file transaksi utama, dan fungsi-fungsi pemrosesan sederhana.
2. Rancang bangun *prototype* dengan bantuan *software* seperti *word  
   processor, spreadsheet, database,* pengolah grafik, dan *software* *CASE* (*Computer-Aided System Engineering*).
3. Uji *prototype* untuk memastikan *prototype* dapat dengan mudah dijalankan untuk tujuan demonstrasi.
4. Siapkan *prototype* USD (*User’s System Diagram*) untuk mengidentifikasi bagian-bagian dari perangkat lunak yang di*prototype*kan.
5. Evaluasi dengan pengguna untuk mengevaluasi *prototype* dan melakukan perubahan jika diperlukan.
6. Transformasikan *prototype* menjadi perangkat lunak yang beroperasi penuh dengan melakukan penghilangan kode-kode yang tidak dibutuhkan.



**Sumber : (*Aunur R. Mulyanto, 2008: 22***)

**Gambar 2.6 Tahapan-tahapan *prototyping model***

**2.3 Business Intelligence**

**2.3.1 Peengertian Business Inteligence**

*Business Intelligence* adalah suatu cara untuk mengumpulkan, menyimpan, mengorganisasikan, membentuk ulang, meringkas data serta menyediakan informasi, baik berupa data aktivitas bisnis internal perusahaan, maupun data aktivitas bisnis eksternal perusahaan termasuk aktivitas bisnis para pesaing yang mudah diakses serta dianalisis untuk berbagai kegiatan manajemen (Ricky Akbar, Dkk, 2017 ).

*Business Intelligence* merupakan sebuah proses untuk melakukan *ekstraksi* data-data operasional perusahaan dan mengumpulkannya dalam sebuah data *warehouse*. Selama proses ekstraksi juga dapat dilakukan transformasi dengan menerapkan berbagai formula, agregasi, maupun *validasi* sehingga didapat data yang sesuai dengan kepentingan analisis bisnis. Selanjutnya data di data warehouse diproses menggunakan berbagai analisis statistik dalam proses data mining, sehingga didapat berbagai kecenderungan atau pattern dari data. Hasil penyederhanaan dan peringkasan ini disajikan kepada end user yang biasanya merupakan pengambil keputusan bisnis. Dengan demikian manajemen dapat mengambil keputusan berdasarkan fakta-fakta aktual, dan tidak hanya mengandalkan intuisi dan pengalaman kuantitatif saja ( Imelda, S.T, M.T. 2013 ).

*Business Intelligence* menjelaskan tentang suatu konsep dan metode bagiamana untuk meningkatkan kualitas pengambilan keputusan bisnis berdasarkan sistem yang berbasiskan data. *BI* seringkali dipersamakan sebagaimana *briefing books, report* and *query tools*, dan sistem informasi eksekutif. *BI* merupakan sistem pendukung pengambilan keputusan yang berbasiskan data-data.

*Business Intelligence* adalah alat analisis yang digunakan untuk mengkonsolidasikan data, menganalisis, menyimpan dan mengakses banyak data untuk membantu dalam pembuatan keputusan, seperti perangkat lunak untuk *query database* dan pelaporan, alat untuk analisis data *multi dimensi* , dan data mining

ada beberapa bagian dalam solusi *business intelligence* yaitu, keseluruhan proses dalam *business intelligence* dapat diterjemahkan menjadi langkah-langkah dibawah ini :

1. Identifikasi masalah bisnis yang perlu diselesaikan dengan gudang data dan menentukan data yang diperlukan untuk menyelesaikan masalah tersebut.

2. Identifikasi lokasi dari data-data yang diperlukan dan mengambilnya dari sumber penyimpanannya.

3. Merubah data yang diperoleh dari beragam sumber tersebut ke dalam sebuah data yang konsisten.

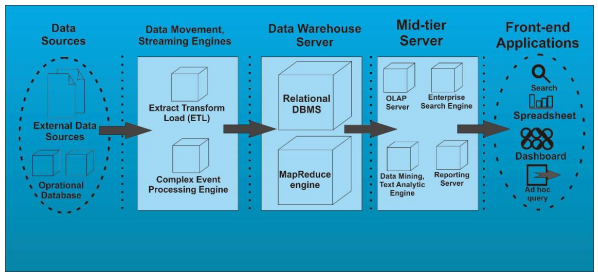
4. Mengambil data yang telah dirubah tersebut ke dalam lokasi yang yang ter*sentralisasi*.

5. Membuat sebuah gudang data dengan data yang ada dalam lokasi yang ter*sentralisasi* tersebut

6. Memasang sebuah produk atau aplikasi yang dapat memberikan akses ke data yang ada dalam *cube* tadi. Ada berbagai macam jalan dan cara untuk berbagai macam tipe pekerjaan ketika berurusan dengan *cube.*

**2.3.2 Arsitektur *Business Intelligence***

Arsitektur dan infrastruktur *business intelligence* telah digambarkan dan disajikan dalam banyak cara yang berbeda dalam literatur teknologi informasi saat ini. Beberapa menggunakan pandangan tradisional arsitektur perangkat lunak dan menjelaskan sistem perangkat lunak, perangkat keras, *middleware, application suites,* data *warehouses* dan transaksi bisnis. Standar *business intelligence* digambarkan sebagai sebuah piramida *BI* yang menunjukkan bagaimana mendistribusikan alat BI yang berbeda untuk kelompok pengguna yang berbeda.

Arsitektur *business intelegence* digambarkan di bawah ini :

**Sumber : (Rekha, 2015)**

**Gambar 2.7 Standarisasi Arsitektur *Business Intelligence***

Adapun penjelasan masing-masing komponen dan tahapan pada arsitektur *business intelligence* adalah sebagai berikut:

1. Data *Sources* (Sumber Data)

Bagian ini berfungsi untuk pengumpulan data dan mengintegrasikan data yang disimpan dari berbagai sumber primer dan sekunder, data tersebut merupakan data yang dimiliki oleh sistem operasional tetapi juga dapat mencakup dokumen yang tidak struktur seperti email dan data yang diterima dari penyedia eksternal, maka dari itu secara garis besar hal ini diperlukan untuk menyatukan dan mengintegrasikan data dari sumber yang berbeda.

1. Data *Movement* dan *Streaming Engines*

Data *movement* dan *Streaming* *Engines* merupakan data yang menampilkan data dari setiap variasi data yang terintegrasi merupakan yang didapat dari berbagai sumber. Tugas Data *Movement* dan *Streaming Engines* ditangani sebuah alat berupa *Extract Transform* *Load* (ETL) yang membatu dalam menemukan masalah kualitas data dan memfasilitasi pemuatan data dengan jumlah yang besar kedalam warehouse. Kualitas data tersebut sangatlah penting dalam penilaian *BI,* jika data yang disajikan kepada user tidak lengkap atau tidak konsisten maka bukan hanya gagal tetapi berpengaruh dan menghambat dalam proses pengambilan keputusan.  
Sedangkan *The Complex Event Processing* (CEP) sebuah alat yang pada tingkatan arsitektur fungsinya sama dengan alat *ETL*, tetapi *ETL* digunakan untuk proses batch data dan ketepatan waktu serta tidak kritis, sementara *CEP* digunakan untuk menangani real time atau hampir mendekati data yang *real time*. Dalam beberapa kasus, *CEP* bermanfaat untuk memvisualisasikan akses data *real time* untuk mendukung keputusan yang cepat.

1. Data *Warehouse Servers*

Data *warehaouse* adalah pengumpulan data subjek yang berorientasi, terintegrasi, waktu yang bervarian, dan *non-volatile* untuk mendukung proses pengambilan keputusan. Setelah data di ekstrak, terintegrasi dan diperiksa kualitas datanya kemudian dimuat dalam suatu resipositori sentral atau yang disebut *warehouse* yang dikelola oleh satu atau lebih *server.* Data *warehouse* dirancang untuk mengenali setiap topik–topik yang menjadi perhatian utama yang menyangkut dengan bisnis sehingga para pengambil keputusan dapat menganalisis dengan mudah.

1. *Mid-tier servers*

*Mid-tier servers* meyediakan fungsi khusus untuk sekenario *BI* yang berbeda dan masuk dalam *server OLAP,* *Enterprise Search Engines*, Data Mining *Engines* dan *Reporting Servers*. *Server OLAP* yang efisien menyajikan model multidimensi untuk aplikasi *front end* atau langsung ke pengguna. Para pengguna dapat melakukan memilah dan memilih, agregasi, menyaring, menurunkan dan memutar data.

e. *Front-end Application*

*Front-end* application merupakan aplikasi yang digunakan langsung oleh pengguna bisnis dalam membuat keputusan. Sebagai contoh alat-alat dalam komponen *front-end* application adalah *enterprise* portals *for* *searching*, *spread* *sheets* dan aplikasi manajemen kinerja yang memvisualisaikan seperti *dashboard* atau alat-alat yang memungkinkan pengguna teknologi tersebut dapat mengakses dengan mengeksekusi *query adhoc* dan alat yang memvisualisasikan model data. Semakin *fleksibel* alat ini lebih memungkinkan eksplorasi data lebih dinamis dan *menginvestigasi* data dengan cara yang berbeda.  
Penggunaan *Business.*

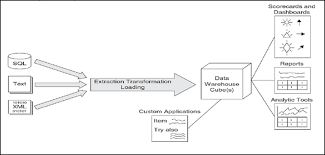
**2.3.3 Fungsi *Business Inteligence(BI)***

*Business intelligence* memiliki fungsi sebagai sistem pendukung pengambilan keputusan dimana sistem dan aplikasi ini mengubah data-data dalam suatu perusahaan atau organisasi ke dalam bentuk pengetahuan. Secara umum, *BI* bertujuan untuk menyajikan berbagai informasi yang disesuaikan dengan butuhan setiap penggunanya. Informasi tersebut dapat berasal dari mana saja, misalnya dari data histori pembelian barang oleh pelanggan, data *histori reparasi*, data *histori komplain*, dan sebagainya. Data-data tersebut kemudian diolah dan disajikan dalam bentuk informasi yang mudah dicerna oleh penggunanya dengan satu tujuan yaitu membantu pencapaian tujuan bisnis perusahaan. *Business Intelligence* (BI) memiliki karakteristik sebagai pendukung ketersediaan data yang relevan yang akandisajikan pada pengguna. Biasanya, *BI* *mengintegrasikan* informasi dari keseluruhan sumber informasi perusahaan sehingga pembuat keputusan dapat membuat analisis dengan berbekal pengetahuan yang lengkap dan *real time* (Imelda, 2013).

**2.3.4 Langkah-Langkah *Business Intelligence***

Menurut Ronald ada beberapa proses dalam *business intelligence* yang diterjemahkan menjadi langkah-langkah dibawah ini (Imelda, 2013) :

1. Identifikasi masalah bisnis yang perlu diselesaikan dengan gudang data dan menentukan data yang diperlukan untuk menyelesaikan masalah tersebut.
2. Identifikasi lokasi dari data-data yang diperlukan dan mengambilnya dari sumber penyimpanannya.
3. Merubah data yang diperoleh dari beragam sumber tersebut ke dalam sebuah data yang konsisten.
4. Mengambil data yang telah dirubah tersebut ke dalam lokasi yang yang *tersentralisasi.*
5. Membuat sebuah gudang data dengan data yang ada dalam lokasi yang *tersentralisasi* tersebut.
6. Memasang sebuah produk atau aplikasi yang dapat memberikan akses ke data yang ada dalam *cube* tadi. Ada berbagai macam jalan dan cara untuk berbagai macam tipe pekerjaan ketika berurusan dengan *cube.*



**Sumber : *(Imelda, 2013)***

**gambar 2.8 Langkah-Langkah *Business Intelligence***

**2.3.5 Manfaat *Business Intelligence***

 Manfaat dan keuntungan yang diperoleh dengan penggunaan *business intelligence* di sebuah organisasi atau perusahaan antara lain adalah sebagai berikut (Turban, Rainer dan Potter, 2011 ) :

1. **Meningkatkan nilai data dan informasi *organisasi***.

Dengan membangun *business intelligence*, maka seluruh data dan informasi dapat *diintegrasikan* sehingga menghasilkan sebuah kesimpulan dari keadaan bisnis yang mudah di akses dan dimengerti sehingga dapat membantu pihak manajerial untuk membuat pengambilan keputusan yang lebih baik.

1. **Memudahkan pengukuran kinerja *organisasi****.*

Dalam mengukur kinerja suatu *organisasi,* sering dipergunakan ukuran yang disebut *Key* *Performance* *Indicator* (KPI). *Business intelligence* dapat dengan mudah menunjukan pencapaian *KPI* suatu *organisasi* dengan mudah, cepat dan tepat. Dengan demikian akan memudahkan pihak-pihak yang terlibat dalam pengambilan keputusan untuk mempersiapkan langkah-langkah antisipasi apabila ada indikator yang menunjukan adanya masalah atau belum tercapainya suatu target.

1. **Meningkatkan nilai investasi teknologi informasi yang sudah ada.**

*Business intelligence* tidak selalu mengubah atau menggantikan sistem informasi yang sudah ada, akan tetapi hanya menambahkan layanan pada sistem-sistem tersebut sehingga data dan informasi dapat di *representasikan* dengan lebih baik.

1. **Meningkatkan efisiensi biaya.**

*Business intelligence* dapat meningkatkan *efesiensi* biaya karena dapat mempercepat seseorang dalam melakukan pekerjaan sehingga menghemat waktu dan mempermudah pemanfaatannya. Waktu yang dibutuhkan untuk mencari data dan mendapatkan informasi yang dibutuhkan semakin singkat dan cara untuk mendapatkannya pun tidak memerlukan pengetahuan yang khusus.

**2.4 *Unified Modeling Language* (UML*)***

**2.4.1 Pengertian *Unified Modeling Language* (UML*)***

*Unified Modeling Language* (UML) adalah “bahasa standar” untuk penulisan cetak biru perangkat lunak. *UML* dapat digunakan untuk *memvisualisasikan,* menentukan, *mengonstruksikan* artifak-artifak suatu sistem *software-intensive*”[Boo05]. Dengan kata lain, sama seperti arsitek bangunan membuat cetak biru untuk digunakan oleh perusahaan kontruksi, arsitek perangkat lunak membuat diagram *UML* untuk membantu pengembangan perangkat lunak membangun perangkat lunak (Pressman Roger S, 2010 ).

Grady Booch, Jim Rumbaugh, dan Ivar Jacobson mengembangkan *UML* pada pertengahan 1990-an dengan banyak umpan balik dari komunitas pengembangan perangkat lunak. *UML* menggabungkan sejumlah notasi pemodelan yang saling bersaing, yang digunakan oleh industri perangkat lunak saat itu. Pada 1997, *UML* 1.0 diserahkan kepada *Object Management Group*, suatu konsorsium nirlaba yang aktif dalam pemeliharaan spesifikasi-spesifikasi untuk digunakan oleh industri komputer. *UML* 1.0 direvisi ke *UML* 1.1 dan diadopsi pada tahun itu juga. Standar sekarang adalah *UML* 2.0 dan sekarang standar *ISO*. Karena standar ini baru, banyak referensi-referensi lama, seperti [Gam95], tidak menggunakan notasi *UML* (Pressman Roger S, 2010 ).

**2.4.2 Diagram UML**

Ada banyak diagram di dalam UML *(Unified Modeling Language),* penulis akan membahas diagram yang digunakan dalam melakukan analisa dan rancangan sistem pengadaan barang sebagai berikut ( Deni Mahdiana, 2011 ).

1. *Use Case Diagram*

*Use Case Diagram* adalah diagram yang menggambarkan kebutuhan sistem dari sudut pandang *user*, yang memperlihatkan hubungan-hubungan yang terjadi antara *actors* dengan *use case* dalam sistem.

Berikut adalah simbol-simbol yang ada pada *Use Case Diagram*:

**Tabel 2.1 Simbol-Simbol yang digunakan pada *Use Case Diagram***

|  |  |  |
| --- | --- | --- |
| **No** | **Simbol** | **Keterangan** |
| 1 | *Use case* | *Fungsionalitas* yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. |
| 2 | Aktor / *Actor*  *actor* | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. |
| 3 | Asosiasi / *Association* | Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor. |
| 4 | Ekstensi / *Extend* | Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan. |
| 5 | Generalisasi / *Generalization* | Hubungan generalisasi dan *spesialisasi* (umum - khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya. |

**Sumber : (*Rosa A. S dan M. Shalahuddin, 2014* )**

1. *Class Diagram*

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

Berikut adalah simbol-simbol yang ada pada *Class Diagram* :

**Tabel 2.2 simbol-simbol yang digunakan pada *Class Diagram***

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| Kelas  **nama­­\_kelas­**  +operasi()  +atribut | Kelas pada struktur sistem. |
| Antarmuka / *Interface* | Sama dengan kosep *interface* dalam pemrograman berorientasi objek. |
| Asosiasi / *Association* | Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*. |
| Asosiasi berarah / *Directed Association* | Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*. |
| Generalisasi | Relasi antar kelas dengan makna generalisasi-spealisasi (umum khusus). |
| Kebergantungan / *Dependency* | Relasi antar kelas dengan makna kebergantungan antar kelas. |
| Agregesi / *Aggregation* | Relasi antar kelas dengan makna semua-bagian (*whole-port*). |

**Sumber : (*Rosa A. S dan M. Shalahuddin, 2014* )**

1. *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem atau *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antar muka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu ditampilkan pada perangkat lunak.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

**Tabel 2.3 simbol-simbol yang digunakan pada *Activity Diagram***

|  |  |
| --- | --- |
| **Simbol** | **Keterangan** |
| Status awal | Status awal aktifitas sistem, sebuah diagram aktifitas memiliki sebuah status awal. |
| aktivitas | Aktivitas yang dilakukan *system*, biasanya diawali dengan kata kerja. |
| Percabangan / *Decision* | Asssosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. |
| Penggabungan / *Join* | Assosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu. |
| Status akhir | Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir. |
| *Swimlane*    Atau | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi. |

**Sumber : (*Rosa A. S dan M. Shalahuddin, 2014 )***

1. *Sequence Diagram*

*Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display* dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence Diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence Diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan (Deni Mahdiana, 2011

Berikut ini adalah simbol-simbol yang ada pada *Sequence Diagram*:

**Tabel 2.4 simbol-simbol yang digunakan pada *Sequence Diagram***

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| *Actor*  Nama actor atau  Nama aktor  Tanpa waktu aktif | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari actor adalah orang, tapi actor belum tentu merupakan orang biasanya  Dinyatakan menggunakan kata benda di awal *fase* nama aktor. |
| Garis hidup / *Lifeline* | Menyatakan kehidupan suatu objek |
| Objek  Namaobjek : nama kelas | Menyatakan objek yang berinteraksi pesan |
| Waktu aktif | Menyatakan objek dalam keadaan aktif dan berinteraksi pesan |
| Pesan tipe *create*  <<*create*>> | Status akhir yang dilakukansistem, sebuah diagram aktivitas memiliki status akhir. |
| Pesan tipe *call*  1 : nama\_metode() | Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri.  1 : nama\_metode() |
| Pesan tipe *send*  1: masukan | Menyatakan suatu objek bahwa mengirim data, masukan dan informasi ke objek lain, arah panah mengarah pada objek yang dikirim |
| Pesan tipe *return*  1: keluaran  ---------------------- | Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang dikirim. |
| Pesan tipe *destroy*  <<*destroy*>> | Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*. |

**Sumber : (*Rosa A. S dan M. Shalahuddin, 2014* )**

1. *Statechart Diagram*

*Statechart Diagram* digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau objek. Jika *Sequence Diagram* digunakan untuk interaksi antar objek maka *Statechart Diagram* digunakan untuk interaksi didalam sebuah objek (Rosa A. S dan M. Shalahuddin, 2014 ).

**Tabel 2.5 simbol-simbol yang digunakan pada *Statechart Diagram***

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| *Start* / Status Awal (Initial State) | *Start* atau *initial state* adalah *state* atau keadaan awal pada saat sistem mulai hidup. |
| *End* /Status Akhir (*Final State*) | *End* atau *final state* adalah *state* keadaan akhir dari daur hidup suatu sistem. |
| *Event*  *Event* | *Event* adalah kegiatan yang menyebabkan berubahnya status mesin. |
| *State*  *state* | Sistem pada waktu tertentu. *State* dapat berubah jika ada *event* tertentu yang memicu perubahan tersebut. |

**Sumber : (*Rosa A. S dan M. Shalahuddin, 2014* )**

1. *Deployment Diagram*

*Deployment Diagram* menunjukkan susunan fisik sebuah sistem, menunjukkan bagian perangkat lunak mana yang berjalan pada perangkat keras mana. Hal utama dalam diagram ini adalah pusat-pusat yang dihubungkan oleh jalur komunikasi. Sebuah pusat adalah sebuah titik yang dapat mengumpulkan beberapa perangkat lunak. Pusat-pusat mempunyai dua bentuk. Sebuah alat adalah perangakat keras *( hardware),* alat ini berupa sebuah komputer atau perangkat keras yang lebih sederhana yang berhubungan pada sebuah sistem.

**Tabel 2.6 simbol-simbol yang digunakan pada *Deployment Diagram***

|  |  |
| --- | --- |
| **Simbol** | **Keterangan** |
| *Package*  *package* | *Package* merupakan sebuah bungkusan dari satu atau lebih *node.* |
| *Node*  Nama\_*node* | Biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika di dalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang di ikut sertakan harus sesuai dengan komponen yang telah di defenisikan sebelumnya pada diagram komponen. |
| Kebergantungan / *dependency* | Kebergantungan antar *node*, arah panah mengarah pada *node* dipakai |
| *Link* | Relasi antar *node* |

**Sumber : (*Rosa A. S dan M. Shalahuddin, 2014*)**

1. *Communication Diagram*

*Diagram komunikasi* mengelompokkan message pada kumpulan diagram sekuen menjadi sebuah diagram. Dalam diagram komunikasi yang dituliskan adalah operasi/metode yang dijalankan antar objek yang satu dan objek lainnya secara keseluruhan, oleh karena itu dapat diambil dari jalannya interaksi pada semua diagram sekuen. Penomororan metode dapat dilakukan berdasarkan urutan dijalankannya metode atau operasi diantara objek yang satu dengan objek lainnya atau objek itu sendiri.

Berikut adalah simbol-simbol yang ada pada diagram kolaborasi:

**Tabel 2.7 simbol-simbol yang digunakan pada *Communication Diagram :***

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| Objek  Nama\_objek : nama\_kelas | Objek yang melakukan interaksi pesan |
| Link | Relasi antar objek yang menghubungkan objek satu dengan yang lainya atau dengan dirinya sendiri  Nama\_objek : nama\_kelas |
| Arah pesan / *stimulus* | Arah pesan yang terjadi,jika pada suatu *link* ada dua arah pesan yang berbeda maka arah juga digambarkan dua arah pada dua sisi *link* |

**sumber : (*Rosa A. S dan M. Shalahuddin, 2014*)**

**2.5 PHP *(Pre-Hypertext Prepocessor)***

**2.5.1 Sejarah PHP *(Pre-Hypertext Prepocessor)***

*PHP Hypertext Preprocessor* atau sering disebut *PHP* merupakan bahasa pemograman berbasis *server-side* yang dapat melakukan parsing *script php* menjadi *script* web sehingga dari sisi *client* menghasilkan suatu tampilan yang menarik. *PHP* merupakan pengembangan dari *F1* atau *Form Interface* yang dibuat oleh Rasmus Lerdoff pada tahun 1995.

Beberbeda dengan *HTML*, kode *PHP* tidak diberikan secara langsung oleh *server* ketika ada permintaan atau *request* dari sisi client namun dengan cara pemrosesan dari sisi *server*. Kode *PHP* sering kali digabungkan dengan kode *HTML*. Untuk membedakannya dengan *HTML*, setiap kode *PHP* ditulis selalu diberi tag pembuka yaitu *<?php* dan pada akhir kode *PHP* diberi tag penutup yaitu *?>.*

**2.5.2 Pengertian PHP**

*PHP* adalah bahasa skrip yang dapat ditanamkan atau disisipkan kedalam *HTML*. *PHP* banyak dipakai untuk membuat situs web dinamis. *PHP* dapat digunakan untuk membangun sebuah *CMS(Content Management System).* File *PHP* dapat berisi teks, *HTML, CSS, Java Script*, dan kode *PHP*. Hal yang bisa dilakukan *PHP* adalah dapat menghasilkan konten halaman dinamis. Dapat membuat, membuka, membaca, menulis, menghapus dan menutup file pada server (Eka Praja Wiyata Mandala, S.kom., M.kom,2015 ).

Fungsi *PHP* yang digunakan untuk Mengolah File *Text* menurut (Welly Widodo Sindu Putra, dkk, 2010 ).

1. *copy (string source, string dest),* digunakan untuk menggandakan file.
2. *is\_dir (string dirname),* digunakan untuk memastikan bahwa direktori yang ditunjuk ada.
3. *opendir (string path),* digunakan untuk membuka direktori dan menjadikannya sebuah resource yang dapat dibaca.
4. readdir *(resource dir\_handle),* digunakan untuk membaca isi dari *resource direktori* yang sudah dibuka.
5. *is\_file (string filename),* digunakan untuk memastikan bahwa file yang ditunjuk benar.
6. *file\_exists (string filename),* digunakan untuk memastikan keberadaan sebuah file.
7. *fopen (string filename, string mode [, int use\_include\_path [, resource zcontext]]),* digunakan untuk membuka file text dan menjadikannya sebuah *resource* yang dapat dibaca.
8. *fread (resource handle, int length),* digunakan untuk membaca keseluruhan data pada *resource* dari file text yang dibuka.
9. *fwrite (resource handle, string string [, int length]),* digunakan untuk menuliskan data pada file text yang sudah dibuka.
10. *fputs (resource handle [, int length]),* digunakan untuk menuliskan data pada file text yang sudah dibuka.
11. *fclose (resource handle),* digunakan untuk menutup / menghapus *resource* yang dibuka.
12. *explode (string separator, string string [, int limit]),* digunakan untuk memisahkan data *bertype* string menjadi sebuah *array* dengan menggunakan tanda pemisah tertentu.

**2.5.3 Fungsi dan Kelebihan PHP**

*PHP* memiliki manfaat yang sangat besar bagi para web *programmer* dan web *developer* pada saat membuat *website* keren yang dinamis seperti mambaca file, menulis file, manampilkan gambar, animasi dan *movie*, dan yang paling penting adalah dapat melakukan koneksi terhadap database seperti *MySQL*.

Perlu diketahui bahwa *PHP* merupakan bahasa scripting dari sisi server *(server-side).* Dapat dikatakan secara sederhana bahwa *PHP* juga dapat digunakan bersama dengan *HTML* sehingga web *programmer* dapat membuat sebuah *HTML* yang dinamis atau *website* yang tidak lagi menggunakan berlayer-layer halaman yang panjang.

Banyak sekali kelebihan yang dimiliki *PHP* bila dibandingkan dengan bahasa pemograman yang lain, di antaranya (Yudha Yudhanto, Agus Purbayu, 2014 ):

1. Bisa mebuat *Web* menjadi Dinamis.
2. PHP bersifat *Open Source* yang berarti dapat digunakan oleh siapa saja secara gratis.
3. Program yang dibuat dengan *PHP* bisa dijalanankan oleh semua sistem operasi karena *PHP* berjalan secara *web base* yang artinya semua sistem operasi bahkan gadget mobile apa pun asalkan mempunyai browser dapat menikmati program *PHP*.
4. Aplikasi *PHP* lebih cepat dibandingkan dengan *ASP* maupun *Java*.
5. Mendukung banyak paket *Database* seperti *MysSQL*, *Orwcle*, *PostgreSQL*, dan lain-lan.
6. Bahasa pemograman *PHP* tidak memerlukan *compile* dalam penggunaannya.
7. Banyak *Web Server* yang mendukung *PHP* seperti *Apache, Lighttpd, II*S, dan lain-lain.
8. Pengembangan aplikasi *PHP* sangat cepat karena banyaknya dokumentasi, referensi, dan antar sesama *developer* yang turut membantu dalam pengembangannya.
9. Banyak bertebaran apliaksi dan program *PHP* yang *free*, serta siap digunakan atau dikembangkan sendiri seperti *WordPress*, *Joomla*, *Opencart, PrestaShop, Drupal, Magento,* dan lain-lai.

**2.5.4 Sintak PHP**

“Sebuah sintaks *PHP* dapat ditempatkan dimanapun dalam sebuah dokumen dengan cara menyisipkan tanda <?***php*** untuk memulai dan tanda ?> untuk mengakhiri” (Eka Praja Wiyata Mandala, S.kom., M.kom, 2015 ).

Bentuk umum :

<?php

// Skrip PHP diletakkan disini

?>

Sebuah *file* PHP biasanya mengandung HTML, dan beberapa *script* PHP. Berikut ini, contoh *file* *PHP* sederhana dengan *script* PHP yang menggunakan fungsi PHP bawaan **“echo”** untuk *output teks***“Hello World!”** pada halaman *web.*

<!DOCTYPE html>

<html>

<body>

<h1>Halaman PHP Pertama Saya</h1>

<?php

*Echo “ Hello World”;*

*?>*

*</body>*

</html>

* + 1. **Mendeklarasikan Variabel Dalam PHP**

*Variable* adalah sebuah wadah untuk menyimpan informasi. Dalam *PHP*, dimulai dengan tanda $ yang kemudian diikuti oleh nama *variable*. Sebuah *variable* dapat berupa nama pendek (dalam bentuk huruf) atau nama yang lebih deskriptif (dalam bentuk kata). Cara penulisan *variable* dalam *PHP* tidak bisa sembarangan, harus sesuai dengan cara penulisan yang sudah ditetapkan (Eka Praja Wiyata Mandala, S.kom., M.kom, 2015). Berikut aturan penulisan *variable* dalam *PHP* :

1. *Variable* dimulai dengan tanda $, diikuti dengan nama *variable*.
2. Sebuah nama *variable* harus diawali dengan huruf atau karakter garis bawah.
3. Sebauh nama *variable* tidak boleh dimuali dengan angka.
4. Sebuah nama *variable* hanya dapat berisi karakter *alpha-numerik* dan garis bawah (a-z 0-9, dan \_).

Nama *variabel* adalah *case-sensitive* ($nama dan $NAMA adalah dua variabel yang berbeda).

* 1. ***MySQL***

**2.6.1 Pengertian *MySQL***

*MySQL* adalah program aplikasi *database* yang berbasis *open source. MySQL* mampu menangani *database* yang kompleks dan cukup besar. *MySQL* juga dapat menangani database *client server*. *MySQL* AB membuat MySQL sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual di bawah lisensi komersial untuk kasus-kasus di mana penggunaannya tidak cocok dengan *GPL*. Tidak seperti *Apache* yang merupakan *software* yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, *MySQL* dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia, yaitu *MySQL* *AB*. *MySQL* *AB* memegang penuh hak cipta hamper atas semua kode sumbernya. Orang yang mendirikan *MySQL AB* adalah: David Axmark dan Allan Larsson berasal dari Swedia, dan Michael *“Monty”* Widenius berasal dari Finlandia (Eka Praja Wiyata Mandala, S.kom., M.kom, 2015 ).

MySQL *(My Structured QueryLanguage)* atau yang biasa dibaca mai-se-kuel adalah sebuah program pembuat dan pengelola database atau yang sering disebut dengan *DBMS (DataBase Management System),* sifat dari *DBMS* ini adalah *Open Source*. *MySQL* sebenarnya produk yang berjalan pada *platform Linux*, dengan adanya perkembangan dan banyaknya pengguna, serta lisensi dari database ini adalah *Open Source*, maka para pengembang kemudian merilis versi *Windows.* Selain itu *MySQL* juga merupakan program pengakses *database* yang bersifat jaringan, sehingga dapat digunakan untuk aplikasi *Multi User* (Banyak Pengguna). Kelebihan lain dari *MySQL* adalah menggunakan bahasa *query* (permintaan) *standard* *SQL* *(Structured QueryLanguage).* Sebagai sebuah program penghasil *database*, *MySQL* tidak mungkin berjalan sendiri tanpa adanya sebuah aplikasi pengguna *(interface)* yang berguna sebagai program aplikasi pengakses *database* yang dihasilkan ( Uswatun Hasanah, 2013).

* 1. **Website**

**2.7.1 Pengertian Website**

Website adalah keseluruhan halaman–halaman web yang terdapat dalam sebuah domain yang mengandung informasi. Sebuah *website* biasanya dibangun di atas banyak halaman web yang lainnya disebut dengan *hyperlink*. Sedangkan teks yang dijadikan media penghubung disebut *hypertext*.

Domain adalah nama unik yang dimiliki oleh sebuah institusi sehingga bisa diakses melalui internet, misalnya *lintau.com, google.com* dan lain–lain. Untuk mendapatkan sebuah domain kita harus melakukan registrasi pada *register–register* yang ditentukan.

* 1. **Web Server**

**2.8.1 Pengerrtian Web Server**

*Web Server* pada dasarnya adalah perangkat lunak khusus yang bertugas melayani permintaan – permintaan dari *browser* *web* akan dokumen – dokumen yang tersimpan di dalamnya. *Web Server* merupakan server internet yang mampu melayani koneksi *transfer* data dalam protocol *HTTP*. *Web Server* merupakan hal yang terpenting dari *server* di *internet* dibandingkan *server* lainya seperti *e-mail* server, *FTP server* ataupun *news server*. Hal ini disebabkan *web server* telah dirancang untuk dapat melayani beragam jenis data, dari *text* sampai grafis 3 (tiga) dimensi. Kemapuan ini telah menyebabkan berbagai *institusi* seperti *universitas* maupun perusahaan dapat menerima kehadirannya dan juga sekaligus menggunakannya sebagai sarana di *internet*.

Web server juga dapat menggabungkan dengan dunia *mobile wireless internet* atau yang sering disebut sebagai *WAP* *(Wireless Accsess Protocol),* yang banyak digunakan sebagai sarana *handphone* yang memiliki fitur *WAP*. Dalam kondisi ini, *web server* tidak lagi dilayani data *file HTML* tetapi telah melayani *WML* *(Wireless Markup Language).*

Salah satu *software* yang bisa digunakan oleh banyak *web master* di dunia adalah *apache.* *Software* tersebut dapat kita *download* secara gratis dari *web* resmi *apache*, yaitu *http://www.apache.org.* dalam penggunaannya *Apache* merupakan *software* *open source* yang sekarang ini sudah merebut pasar dunia lebih dari 50%. *Web server* ini *fleksibel* terhadap berbagai *system* operasi seperti *windows9x/NT* ataupun *UNIX/Linux*.

Kelebihan *web server Apache*:

1. *Freeware (software* gratisan)
2. Mudah diinstall.
3. Mampu beroperasi pada berbagai *platform* *system* operasi.
4. Mudah mengkonfigurasikannya.
5. *Apache Web server* mudah dalam menambahkan *peripheral* lainnya ke dalam *platform web servernya*, misalnya: untuk menambahkan modul, cukup hanya *menset* file konfigurasinya agar mengikutsertakan modul itu ke dalam kumpulan modul lain yang sudah dioperasikan.
6. Dapat dijadikan pengganti bagi *NCSA web server*.
7. Perbaikan terhadap kerusakan dan *error* *NCSA* 1.3 dan 1.4
8. Merespon *client* lebih cepat dari pada *server NCSA*.
9. Mampu di kompilasi sesuai dengan spesifikasi *HTTP* yang sekarang.
10. Kita dapat men-set respon *error* yang akan dikirim *web server* dengan menggunakan file atau skrip.
11. Secara otomatis menjalankan file *intex.html*, halaman utamanya, untuk ditampilkan secara otomatis pada *client*nya.
12. Lebih aman karena memiliki level – level pengamanan.
13. Apache mempunyai komponen dasar terbanyak di antara *webserver–webserver* lain, yang berarti bahwa *webserver Apache* termasuk salah satu dari *web server* yang lengkap.
14. Performasi dan konsumsi sumber daya (*resource*) dari *webserver* *apache* tidak terlalu banyak, hanya sebesar 20 *MB* untuk file – file dasarnya dan setiap daemonnya hanya memerlukan sebesar 950 *KB* memory per-*child*.
15. Mendukung transaksi yang aman *(secure transaction)* menggunakan SSL *(Secure Socket Layer).*
16. Mempunyai dukungan teknis melalui *web.*
17. Mempunyai kompatibilitas *platform* yang tinggi.

Beberapa perangkat lunak *server web* mempunyai *feature* seperti *server-sideprogramming*, *security control* dan lain sebagainya. Meskipun beragam macamnya, secara fungsional semua jenis *server web* adalah sama saja, yaitu berfungsi melayani permintaan-permintaan dari *browser* web.