# BAB II
# TINJAUAN PUSTAKA

## II.1. Profil Tempat Kerja Praktek

Tinjauan umum perusahaan praktek kerja lapangan pada bab ini penulis menjelaskan secara singkat profil Balai Pelatihan Pendidik dan Tenaga Kependidikan Pendidikan Kejuruan (BPPTKPK) Dinas Provinsi Jawa Barat.

### II.1.1. Sejarah Singkat Perusahaan

Balai Pelatihan Pendidik dan Tenaga Kependidikan Pendidikan Kejuruan (BPPTKPK) secara fisik berdiri pada tahun 1975 dengan nama Pusat Latihan Pendidikan Teknik (PLPT). Tahun 1975 s.d. 1978 PLPT dipimpin oleh Drs. M. Bakrie, MA. Tahun 1978 PLPT berganti nama menjadi Balai Latihan Pendidikan Teknik (BLPT). BLPT mengalami 6 (enam) kepemimpinan yaitu : (1) Drs. M. Bakrie, MA. tahun 1978 s.d. 1981, (2) Drs. Djanakum tahun 1981 s.d. 1983, (3) Drs. Daslam Karmananjaya tahun 1983 s.d. 1985, (4) Drs. Didikh Suryana, M.Ed. tahun 1985 s.d. 1987, (5) Drs. Rakhmad Dumadi, BE. tahun 1988 s.d. 1994, (6) Drs. H. Supri Raya tahun 1995 s.d. 2001.

Tahun 2002 melalui Keputusan Gubernur Jawa Barat Nomor 39 Tahun 2001 BLPT berubah menjadi Balai Pengembangan Teknologi Pendidikan (BPTP), yang melalui 4 (empat) kepemimpinan, yaitu : (1) Drs. F.H. Abiyanto,MM. tahun 2002 s.d. 2003, (2) Drs. H. Nanang Mustaram,M.Ed

Tahun 2003 s.d. 2005, (3) Drs. H. Otji S. Wiharjadi, M.Pd. tahun 2005 s.d. 2008, (4) Drs. H. Nandang Djunaedi, MM tahun 2009 s.d. 2010.

Tahun 2010 melalui Peraturan Gubernur Jawa Barat No. 113 tahun 2009 BPTP berubah menjadi Balai Pelatihan Pendidik dan Tenaga Kependidikan Pendidikan Kejuruan (BPPTKPK), dipimpin oleh Drs. H. Nandang Djunaedi, MM. hingga sekarang tahun 2010 s.d. sekarang.

## II.2. Visi dan Misi

Balai Pelatihan Pendidik dan Tenaga Kependidikan Pendidikan Kejuruan (BPPTKPK) memiliki visi dan misi, yaitu:

### II.2.1. Visi

"Akselerasi Peningkatan Mutu Pendidikan Kejuruan menuju masyarakat Jawa Barat Bertqwa, Mandiri, Dinamis dan Sejahtera".

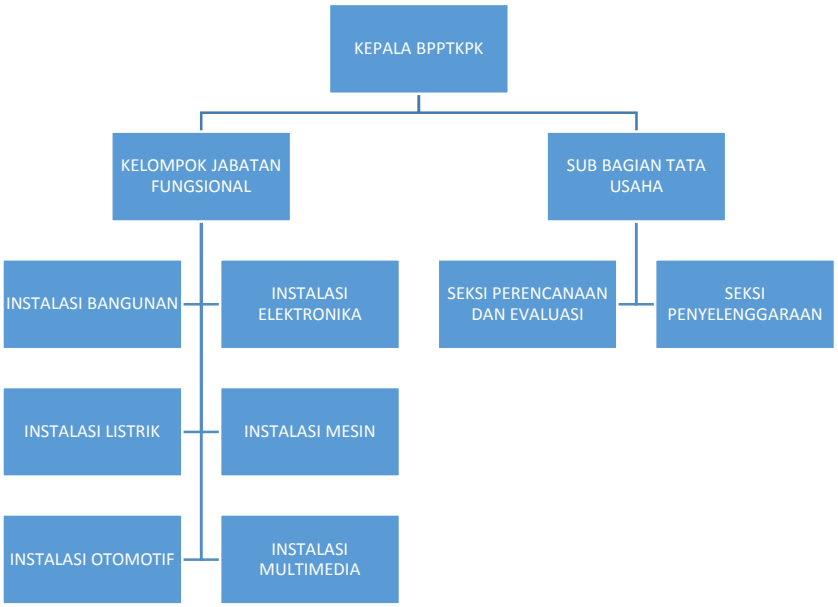
### II.2.2. Misi

**1)** Optimalisasi dan pengembangan sumberdaya kelembagaan dalam upaya meningkatkan layanan pendidikan kejuruan secara produktif, efektif, efisien dan akuntabel.

**2)** Meningkatkan mutu, daya saing dan relevansi pendidikan melalui layanan pelatihan pendidik dan tenaga kependidikan Pendidikan kejuruan yang menguasai teknologi dan berwawasan global.

## II.3. Struktur Organisasi

Organisasi adalah tempat atau wadah orang berkumpul untuk saling bekerjasama untuk mencapai tujuan dan manfaat bersama. Dengan demikian dapat dikatakan bahwa struktur organisasi pada suatu organisasi merupakan kerangka dasar yang menggambarkan alur hubungan antara bagian yang satu dengan yang lainnya, sehingga suatu bagian dalam organisasi tersebut menjadi jelas kedudukan, jabatan, wewenang dan juga tanggung jawabnya.




**Gambar II.1. Struktur Organisasi Balai Pelatihan Pendidik dan Tenaga Kependidikan (BPPTKPK) Dinas Pendidikan Provinsi Jawa Barat**

## II.4. Deskripsi Kerja

Penjelasan mengenai tugas dari masing-masing bidang dapat digambarkan sebagai berikut:

**1) Kepala BPPTKPK**

a) Tugas Pokok

Memimpin, mengkoordinasikan, dan memotivasi serta mengawasi ruang lingkup tugas pokok Balai Pelatihan Pendidik dan Tenaga Kependidikan Pendidikan Kejuruan (BPPTKPK) Dinas Provinsi Jawa Barat dan memberikan keleluasan kepada 26 kabupaten Jawa Barat untuk melakukan pelatihan

b) Fungsi

Kepanjangan tugas dan Dinas Pendidikan Provinsi Jawa Barat untuk pengembangan teknologi umum dan teknologi pendidikan.

**2) Sub Bagian Tata Usaha**

Sub Bagian Tata usaha mempunyai tugas membantu Kepala BPPTKPK di bidang ketatausahaan, kepegawaian, keuangan, perlengkapan umum serta perencanaan dan pelaporan dalam administrator manajemen Balai dan mengawasi struktural dalam pelaksanaan tugas pokok BPPTKPK.

**3) Seksi Penyelenggaraan**

Tugas pokoknya adalah mengumpulkan, menghimpun, menganalisa serta menyusun rencana dan program pelakasanaan pelatihan

**4) Seksi Perencanaan dan Evaluasi**

Tugas pokoknya adalah merencanakan pengembangan program model dan sistem pembelajaran, mengevaluasi, memantau dan membina pemanfaatan teknologi informasi untuk pendidikan serta program perencanaan dalam jangka pendek dan jangka panjang.

**5) Kelompok Jabatan Fungsional**

Kelompok Jabatan Fungsional mempunyai tugas pokok melaksanakan proses pembelajaran kepada seluruh SMK terdiri dari sejumlah jabatan fungsional yang terbagi dalam berbagai kelompok sesuai dengan bidang keahliannya. Kelompok Jabatan fungsional di BPPTKPK ada 6 Jabatan Fungsional dalam data ketenagaan sesuai dengan keahliannya sebagai berikut:

a. Instalasi Bangunan
b. Instalasi Elektronika

c. Instalasi Listrik
d. Instalasi Mesin
e. Instalasi Otomotif
f. Instalasi Multimedia

## II.5. Landasan Teori

Subbab ini berisikan teori - teori pendukung yang digunakan dalam proses analisis dan implementasi pada permasalahan yang ada di BPPTKPK.

### II.5.1. Pengertian Sistem

Sistem adalah suatu jaringan kerja dari dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu. [2]

**1. Elemen Sistem**

Sistem informasi terdiri dari elemen-elemen yang terdiri dari orang, prosedur, perangkat keras, perangkat lunak, basis data, jaringan komputer dan komunikasi data. Semua elemen ini merupakan komponen fisik.

a. Orang

Orang atau personil yang dimaksudkan yaitu operator komputer, analisis sistem, programer, personil data entry dan manajer sistem informasi

b. Prosedur

Prosedur merupakan elemen fisik. Hal ini di sebabkan karena prosedur disediakan dalam bentuk fisik seperti buku panduan dan instruksi. Ada 3 jenis prosedur yang dibutuhkan, yaiutu instruksi untuk pemakai, instruksi untuk penyiapan masukan, instruksi pengoperasian untuk karyawan pusat komputer.

c. Perangkat keras

Perangkat keras bagi suatu sistem informasi terdiri atas komputer (pusat pengolah, unit masukan/keluaran), peralatan penyiapan data dan terminal masukan/keluaran.

d. Perangkat Lunak

Perangkat lunak dapat dibagi dalam tiga jenis utama:

1) Sistem perangkat umum, seperti sistem pengoperasian dan sistem manajemen data yang memungkinkan pengoperasian sistem komputer.
2) Aplikasi perangkat lunak umum, seperti model analisis dan keputusan.

3) Aplikasi perangkat lunak yang terdiri atas program yang secara spesifik dibuat untuk setiap aplikasi.

e. Basis data

File yang berisi program dan data dibuktikan dengan adnya media penyimpanan secara fisik, seperti diskette, harddisk, *magnetic tape* dan sebagainya. File juga meliputi keluaran tercetak dan catatan lain diatas kertas, mikro film dan lain sebagainya.

f. Jaringan Komputer

Jaringan komputer adalah sebuah kumpulan komputer, printer dan peralatan lainya yang terhubung dalam satu kesatuan. Informasi dan data bergerak melalui kabel-kabel atau tanpa kabel sehingga memungknkan pengguna jaringan komputer dapat saling bertukar dokumen dan data.

g. Komunikasi Data

Komunikasi data merupakan bagian dari telekomunikasi yang secara khusus berkenaan dengan transmisi atau pemindahan data dan informasi diantara komputer dan piranti-piranti yang lain dalam bentuk digital yang dikirimkan melalui media komunikasi data. Data berarti informasi yang disajikan oleh isyarat digital. Komunikasi data merupakan bagian vital dari suatu sistem informasi karena sistem ini menyediakan infrastruktur yang memungkinkan komputer dapat berkomunikasi satu sama lain.

**2. Karakteristik Sistem**

Suatu sistem mempunyai karakteristik atau sifat tertentu, yaitu mempunyai: [2]

a. Komponen.(*Components*)

Komponen terdiri dari sejumlah komponen yang saling berinteraksi dan bekerja sama membentuk satu kesatuan.

b. Batas Sistem (*Boundary*)

Batas sistem merupakan daerah yang membatasi antara sistem dengan sistem lainya atau dengan lingkungan luarnya.

c. Lingkungan Luar Sistem (E*nvironments*)

Lingkungan luar sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem.

d. Penghubung Sistem (*Interface*)

Penghubung sistem merupakan media penghubung antara subsistem, yang memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem

lainnya.

e. Masukan sistem (*input*)

Masukan sistem adalah energi yang dimasukkan kedalam sistem, yang dapat berupa masukkan perawatan(Maintenance input) dan masukkan signal (signal input).

f. Keluaran sistem (*output*)

Keluaran sistem adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

g. Pengolahan sistem (*process*)

Suatu sistem dapat mempunyai suatu bagian pengolah yang akan merubah masukkan menjadi keluaran.

h. Sasaran atau Tujuan Sistem

Suatu system mempunyai tujuan atau sasaran, kalau sistem tidak mempunyai sasaran maka sistem tidak akan ada. Suatu sistem dikatakan berhasil apabila mengenal sasaran atau tujuannya karena sasaran sangat berpengaruh pada masukan dan keluaran sistem yang dihasilkan.

**3. Klasifikasi Sistem**

Sistem dapat diklasifikasikan dari beberapa sudut pandang, diantaranya sebagai berikut ini: [2]

a. Sistem diklasifikasikan sebagai sistem abstrak (*abstract system*) dan sistem fisik (*physical system*) Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik. Misalnya sistem teologia, yaitu sistem yang berupa pemikiran-pemikiran hubungan antara manusia dengan Tuhan. Sistem fisik merupakan sistem yang ada secara fisik. Misalnya sistem komputer, sistem akuntansi, sistem produksi dan lain sebagainya.

b. Sistem diklasifikasikan sebagai sistem alamiah (*natural system*) dan sistem buatan manusia (*human made system*) Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat manusia. Misalnya sistem perputaran bumi. Sistem buatan manusia adalah sistem yang dirancang oleh manusia. Sistem buatan manusia yang melibatkan interaksi antara manusia dengan mesin disebut dengan *human-machine* sistem atau ada yang menyebut dengan *man-machine system*. Sistem informasi merupakan contoh *man-machine system*, karena menyangkut penggunaan komputer yang berinteraksi dengan manusia.

c. Sistem diklasifikasikan sebagai sistem tertentu (*deterministic system*) dan sistem tidak tentu (*probabilistic system*) Sistem tertentu beroperasi dengan tingkah laku yang sudah dapat diprediksi. Interaksi diantara bagian-bagiannya dapat dideteksi dengan pasti, sehingga keluaran dari sistem dapat diramalkan. Sistem komputer adalah contoh dari sistem tertentu yang tingkah lakunya dapat dipastikan berdasarkan program-program yang dijalankan. Sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.
d. Sistem diklasifikasikan sebagai sistem tertutup (*closed system*) dan sistem terbuka (*open system*). Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa adanya turut campur tangan dari pihak diluarnya. Secara teoritis sistem tertutup ini ada, tetapi kenyataannya tidak ada sistem yang benar-benar tertutup, yang ada hanyalah *relatively closed system* (secara relatif tertutup, tidak benar-benar tertutup). Sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk lingkungan luar atau subsistem yang lainnya. Karena sistem sifatnya terbuka dan terpengaruh oleh lingkungan luarnya, maka suatu sistem harus mempunyai suatu sistem pengendalian yang baik. Sistem yang baik harus dirancang sedemikian rupa, sehingga secara relatif tertutup karena sistem tertutup akan bekerja secara otomatis dan terbuka hanya untuk pengaruh yang baik saja.

Suatu sistem yang dihubungkan dengan lingkungannya melalui arus sumber daya disebut sistem terbuka. Sebuah sistem pemanas atau pendingin ruangan, contohnya, mendapatkan input-nya dari perusahaan listrik, dan menyediakan panas/dinginnya bagi ruangan yang ditempatinya. Dengan menggunakan logika yang sama, suatu sistem yang tidak dihubungkan dengan lingkungannya adalah sistem tertutup. Sebagai contohnya, sistem tertutup hanya terdapat pada situasi laboratorium yang dikontrol ketat.

### II.5.2. Pengertian Informasi

Secara singkat informasi adalah data yang telah diolah menjadi bentuk yang berarti bagi penerimanya dan berguna untuk pengambilan keputusan saat ini atau di masa mendatang. Akan tetapi secara lengkap informasi bisa disebut sebagai hasil dari pengolahan data dalam suatu bentuk yang lebih berguna dan lebih berarti bagi penirimanya yang menggambarkan suatu kejadian-kejadian yang nyata yang dapat digunakan untuk pengambilan keputusan. [2]

Dari kedua pengertian diatas dapat disimpulkan bahwa Informasi adalah data yang sudah diproses atau diolah sehingga mempunyai nilai bagi penerimanya dan dapat digunakan untuk dasar pengambilan keputusan yang disampaikan melalui media kertas (*hardcopy*), tampilan (*display)* atau suara (*audio*).

### II.5.3. Pengertian Sistem Informasi

Sistem informasi adalah kumpulan informasi di dalam sebuah basis data menggunakan model dan media teknologi informasi digunakan di dalam pengambilan keputusan bisnis sebuah organisasi. [3] Di dalam suatu organisasi, informasi merupakan sesuatu yang penting di dalam mendukung proses pengambilan keputusan oleh pihak manajemen. Sistem ini memanfaatkan perangkat keras dan perangkat lunak komputer, prosedur manual, model manajemen dan basis data. Sistem informasi memiliki komponen berupa subsistem yang merupakan elemen-elemen yang lebih kecil yang membentuk sistem informasi tersebut misalnya bagian input, proses, output. Dimana input itu sendiri adalah sekumpulan data yang akan kita olah menjadi sebuah informasi yang nantinya akan kita sajikan bagi masyarakat, sedangkan proses adalah suatu kegiatan dimana kita mengolah seluruh data yang ada untuk menghasilkan suatu informasi dan Output adalah sekumpulan informasi yang dapat dengan mudah diperoleh, dimengerti dan dimanfaatkan oleh masyarakat. Tanpa ketiga itu sistem informasi tidak dapat berjalan dengan baik.

### II.5.4. Pengertian Inventarisasi Barang

Inventarisasi barang adalah kegiatan melaksanakan pengurusan berupa penyelenggaraan, pengaturan, pencatatan barang-brang, menyusun daftar barang yang bersangkutan ke dalam suatu daftar inventaris barang secara teratur dan menurut ketentuan yang berlaku. [4]

### II.5.5. Tujuan Inventarisasi Barang

Tujuan umum dilakukan inventarisasi adalah dalam rangka usaha penyempurnaan pengurusan dan pengawasan yang efektif terhadap barang-barang milik negara atau swasta. Adapun tujuan khususnya adalah: [4]

1. Untuk menjaga ketertiban administrasi barang yang dimiliki
2. Untuk menghemat keuangan
3. Sebagai Bahan pedoman untuk menghitung kekayaan
4. Untuk Memudahkan pengawasan dan pengendalian barang.

### II.5.6. Fungsi Inventarisasi Barang

Fungsi inventarisasi barang adalah: [4]

1. Menyediakan data dan informasi dalam rangka menentukan kebutuhan dan menyusun rencana kebutuhan barang.
2. Memberikan data dan informasi untuk dijadikan bahan / pedoman dalam pengarahan pengadaan barang.
3. Memberikan data dan informasi untuk dijadikan bahan/pedoman dalam penyaluran barang.
4. Memberikan data dan informasi dalam.
5. Menentukan keadaan barang (barang yang rusak/tua) sebagai dasar untuk menetapkan penghapusannya.
6. Memberikan data dan informasi dalam rangka memudahkan pengawasan dan pengendalian barang.

### II.5.7. Konsep Dasar Analisis Sistem

Adapun konser dasar analisis sistem adalah sebagai berikut: [5]

#### II.5.7.1. OOP (Object Oriented Programming)

OOP (*Object Oriented Programming*) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data

tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri. Pemrograman orientasi-objek menekankan konsep berikut:

### II.5.7.2. Kelas (*Class*)

Kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh '*Class of dog*' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah *Class* adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. Sebuah *Class* secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah *Class* sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

### II.5.7.3. Objek (*Object*)

Membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

### II.5.7.4. Abstraksi (*Abstract*)

Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat 23 melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

### II.5.7.5. Enkapsulasi (*Encapsulation*)

Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek

lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

### II.5.7.6. Polimorfisme (*Polymorphism*)

Melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan. Metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tunggal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

### II.5.7.7. Inheritas (*Inheritance*)

Mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa harus mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).

### II.5.7.8. UML (*Unified Modelling Language*)

UML (*Unified Modeling Language*) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem. Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP.

UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari IBM. UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. 25 UML dikembangkan sebagai suatu alat untuk

analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan system. UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu: [6]

1. ***Use Case Diagram***

*Use case diagram* adalah gambaran graphical dari beberapa atau semua Aktor, use-case, dan interaksi diantara komponen-komponen tersebut yang memperkenalkan suatu sistem yang akan dibangun. Use-case diagram menjelaskan manfaat suatu sistem jika dilihat menurut pandangan orang yang berada di luar sistem. Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar.

Use-case diagram dapat digunakan selama proses analisis untuk menangkap requirement sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, *use case diagram* berperan untuk menetapkan perilaku (*behavior*) sistem saat diimplementasikan. Dalam sebuah model mungkin terdapat satu atau beberapa use-case diagram. Kebutuhan atau *requirements system* adalah fungsionalitas apa yang harus disediakan oleh sistem kemudian didokumentasikan pada model use-case yang menggambarkan fungsi sistem yang diharapkan (*use case*), dan yang mengelilinginya (*Aktor*), serta hubungan antara Aktor dengan use case (*use case diagram*) itu sendiri.

2. ***Conceptual Diagram***

Sebuah diagram konseptual merupakan representasi visual dari cara di mana konsep-konsep abstrak terkait. Hal ini digunakan sebagai bantuan dalam memvisualisasikan proses atau sistem tingkat tinggi melalui serangkaian garis yang unik dan bagan. Diagram konseptual secara luas digunakan dalam segala bidang seperti bisnis, ilmu pengetahuan, dan manufaktur.

3. ***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 atar 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.

**4. *Collaboration Diagram***

*Collaboration diagram* yaitu diagram yang mengelompokkan pesan pada kumpulan diagram sekuen menjadi sebuah diagram. Dalam diagram tersebut terdapat method yang dijalankan antara objek yang satu dan objek lainnya. Di diagram kolaborasi ini, objek harus melakukan sinkronisasi pesan dengan serangkaian pesan-pesan lainnya. *Collaboration Diagram* lebih menekankan kepada peran setiap objek dan bukan pada waktu penyampaian pesan.

**5. *State Diagram***

*State Diagram* adalah diagram untuk menggambarkan behavior, yaitu perubahan state di suatu *Class* berdasarkan event dan pesan yang dikirimkan dan diterima oleh *Class* tersebut. Setiap diagram state hanya boleh memiliki satu start state (*initial state*) dan boleh memiliki satu atau lebih dari satu stop states (*final state*).

**6. *Activity Diagram***

*Activity Diagram* memiliki pengertian yaitu lebih fokus kepada menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dipakai pada business modeling untuk memperlihatkan urutan aktifitas proses bisnis. Memiliki struktur diagram yang mirip *flowchart* atau data *flow diagram* pada perancangan terstruktur. Memiliki pula manfaat yaitu apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. Dan activity dibuat berdasarkan sebuah atau beberapa *use case* pada *use case diagram*.

**7. *Class Diagram***

*Class diagram* adalah sebuah *Class* yang menggambarkan struktur dan penjelasan *Class*, paket, dan objek serta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class diagram* juga menjelaskan hubungan antar *Class* dalam sebuah sistem yang sedang dibuat dan bagaimana caranya agar mereka saling berkolaborasi untuk mencapai sebuah tujuan.

**8. *Object Diagram***

*Objek diagram* adalah diagram yang memberikan gambaran struktur model sebuah sistem, dalam kurun waktu tertentu. Diagram objek yang berasal dari diagram kelas sehingga diagram objek tergantung pada diagram kelas. Objek Diagram, kadang-kadang disebut sebagai *diagram instance* sangat mirip dengan diagram kelas. Seperti diagram kelas *object diagram* juga menunjukkan hubungan antara obyek, tetapi *object diagram* menggunakan contoh-contoh dunia nyata. *Object diagram* digunakan untuk menunjukkan bagaimana sistem akan terlihat seperti pada waktu tertentu. Karena ada data yang tersedia di objek-objek diagram sering digunakan untuk menjelaskan hubungan yang kompleks antara objek.

**9. *Component Diagram***

*Component diagram* adalah diagram UML yang menampilkan komponen dalam system dan hubungan antara mereka. Pada component *View*, akan difokuskan pada organisasi fisik system. Pertama, diputuskan bagaimana kelas-kelas akan diorganisasikan menjadi kode pustaka. Kemudian akan dilihat bagaimana perbedaan antara berkas eksekusi, berkas *dynamic link library* (DDL), dan berkas runtime lainnya dalam system.

**10. *Deployment Diagram***

*Deployment Diagram* adalah diagram yang menggambarkan detail bagaimana komponen disebar kedalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, node, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.

### II.5.7.9. Model *View Controller*

*Model View Controller* (MVC) adalah design pattern atau arsitektur yang digunakan dalam rekayasa perangkat lunak atau aplikasi yang dengan jelas memisahkan antara data (Model) dengan user interface atau tampilan (*View*). Penerapan MVC tidak terbatas pada aplikasi berbasis web. Penggunaan MVC terbukti sangat efektif dalam semua aplikasi. MVC sendiri pertama kali dirancang oleh Trygve Reenskaug dan dipublikasikan pertama kali oleh XEROX PARAC dan Smalltalk sekitar tahun 1970-1980.

Model MVC banyak digunakan karena memiliki banyak keuntungan dalam proses perancangan aplikasi. Beberapa keuntungan adalah:

1. Penggunaan ulang komponen-komponen antarmuka (user interface reusable component).
2. Kemampuan untuk mengembangkan aplikasi dengan antarmuka pengguna secara terpisah.

3. Kemampuan untuk melakukan pewarisan (inheritance) dari berbagai bagian yang berbeda pada suatu hierarki kelas.
4. Kemampuan untuk mendefinisikan kelas-kelas pengaturan tampilan (control style) yang menyediakan fitur-fitur umum secara terpisah dengan fitur-fitur yang akan ditampilkan oleh aplikasi yang dikembangkan.

Dalam implementasinya model MVC memiliki tiga bagian yaitu, memisahkan data (*Model*) dari tampilan (*View*) dan cara bagaimana memprosesnya (*Controller*). Setiap bagian dapat dijelaskan sebagai berikut: [8]

**1. Data (*Model*)**

Pola MVC memiliki layer yang disebut dengan Model yang merepresentasikan data yang digunakan oleh aplikasi sebagaimana proses bisnis yang diasosiasikan terhadapnya. Dengan memilahnya sebagai bagian terpisah, seperti penampungan data, persistence, serta proses manipulasi, terpisah dari bagian lain aplikasi.

Terdapat beberapa kelebihan dalam pendekatan ini. Pertama, membuat detail dari data dan operasinya dapat ditempatkan pada area yang ditentukan (*Model*) dibanding tersebar dalam keseluruhan lingkup aplikasi. Hal ini memberikan keuntungan dalam proses pemeliharaan aplikasi.

Kedua, dengan pemisahan total antara data dengan implementasi interface, komponen model dapat digunakan kembali oleh aplikasi lain yang memiliki kegunaan yang hampir sama.

**2. Tampilan (*View*)**

Layer ini mengandung keseluruhan detail dari implementasi user interface. Disini, komponen grafis menyediakan representasi proses internal aplikasi dan menuntun alur interaksi user terhadap aplikasi. Tidak ada layer lain yang berinteraksi dengan pengguna, hanya *View*.

Penggunaan layer *View* memiliki beberapa kelebihan yaitu yang pertama, memudahkan pengabungan divisi desain dalam development team. Divisi desain dapat berkonsentrasi pada *style, look and feel* dan sebagainya, dalam aplikasi tanpa harus memperhatikan lebih pada detail yang lain.

Dengan memiliki layer *View* yang terpisah memungkinkan ketersediaan multiple interface dalam aplikasi. Jika inti dari aplikasi terletak pada bagian lain (dalam *Model*), multiple

interfaces dapat dibuat (*Swing, Web, Console*) secara keseluruhan memiliki tampilan yang berbeda namun mengeksekusi komponen Model sesuai fungsionalitas yang diharapkan.

### 3. Cara pemprosesan (*Controller*)

Terakhir, arsitektur MVC memiliki layer *Controller*. Layer ini menyediakan detail alur program dan transisi layer, dan juga bertanggungjawab akan penampungan events yang dibuat oleh user dari *View* dan melakukan update terhadap komponen Model menggunakan data yang dimasukkan oleh user.

Kelebihan dalam penggunaan layer *Controller* secara terpisah: Pertama, dengan menggunakan komponen terpisah untuk menampung detail dari transisi layer, komponen *View* dapat didesain tanpa harus memperhatikan bagian lain secara berlebih. Hal ini memudahkan team pengembang multiple interface bekerja secara terpisah dari yang lain secara simultan. Interaksi antar komponen *View* terabstraksi dalam *Controller*.

Kedua, dengan menggunakan layer terpisah yang melakukan update terhadap komponen Model, detail tersebut dihapus dari layer presentasi. Layer presentasi kembali pada fungsi utamanya untuk menampilkan data kepada user. Detail tentang bagaimana data dari user mengubah ketetapan aplikasi disembunyikan oleh *Controller*. Hal ini memisahkan dengan jelas antara *presentation logic* dengan *business logic*.