BAB II
LANDASAN TEORI

2.1. Tinjauan Jurnal

Menurut Hatmoko (2014:59) menyatakan bahwa "Pemanfataan teknologi untuk menunjang sektor pariwisata sangat dibutuhkan agar para wisatawan mudah untuk mengakses infomasi tempat wisata. Perkembangan teknologi internet dan juga perkembangan *smartphone* mendukung pengembangan informasi geografis menggunakan *Global Positioning System* (*GPS*)".Permasalahan yang ada sekarang, diantaranya: Belum banyak petunjuk jalan yang khusus dibuat para wisatawan dalam lingkup yang lebih kecil, yang dapat diakses dimanapun wisatawan berada, baik mengenai lokasi atau mengenai wisata kuliner dan yang lainnya, Pemanfaatan teknologi yang belum maksimal untuk menunjang sektor pariwisata sangat dibutuhkan agar para wisatawan mudah untuk mengakses informasi tempat wisata. Oleh karena itu diusulkan membuat rumusan untuk merancang sistem aplikasi informasi obyek wisata yang dapat digunakan wisatawan sebagai aplikasi informasi obyek wisata secara *real time* menggunakan *GPS* di Pemalang memakai perangkat *mobile* berbasis *android.*

Penelitian ini menggunakan metode *research and development* (R&D), karena penelitian ini merupakan pengembangan dan membuat prototype baru.Dengan rancangan Aplikasi Informasi Obyek Wisata (*Tour Guide*) Secara *Real Time* Menggunakan *GPS* di Bogor pada *platform* android, hasil yang diperoleh adalah Membantu para wisatawan untuk mendapatkan informasi wisata secara efektif dan efisien, Pengguna akan mendapatkan informasi jadwal acara yang akan diselenggarakan di Bogor,Aplikasi dapat berjalan dengan baik pada

platform Android versi 2.2 (*Froyo*) sampai dengan Android versi 4.0 (*Ice Cream Sandwich*).

Menururt Anwar (2015:148) menyatakan bahwa "Perkembangan teknologi komputasi *mobile* telah meningkat pesat. Hal ini ditandai dengan semakin banyaknya fungsi pada perangkat *mobile* tersebut seperti tersedia *Global Positioning System* (GPS) yang telah terintegrasi, dan tersedianya layanan berdasarkan lokasi (*Location Based Service*)". Permasalahan yang ada sekarang adalah Bagaimana membuat aplikasi *mobile* dengan menggunakan teknologi *Location Based Service* (LBS) di *platform* Android, Bagaimana membuat aplikasi yang dapat menampilkan peta dan rute perjalanan menuju lokasi wisata, Bagaimana membuta *database* informasi lokasi wisata yang dapat diakses oleh user melalui mobile device android dan dapat diupdate oleh admin melalui web. Dalam membantu wisatawan dalam menemukan lokasi wisata di kota Semarang, dibangunlah sebuah aplikasi yang dapat memberikan informasi wisata beserta penjelasan singkatnya dan menampilkan peta rute terdekat menuju lokasi wisata yang akan dituju melalui *Google Maps APIs*.

Model analisisnya menggunakan U*se Case, Activity, Class Diagram*, E-R Diagram dan kamus data.Manfaatnya dapat memberikan kemudahan dalam mengetahui letak dan posisi geografis tempat wisata terdekat disekitar pengguna beserta informasi pendukung dengan melalui ponsel Android. Dan dari dapat disimpulkan Sistem dirancang menggunakan pemodelan UML dan metodologi *SDLC,* Sistem ini diimplementasikan menggunakan *script PHP* pada sisi web serta *java* pada aplikasi android. Pada sisi aplikasi android untuk *user* dibangun menggunakan *eclipse Juno, Java JDK7, SDK Rev.20, ADT* dan dibangun untuk

perangkat dengan versi android minimum 2.3 (API 10), dan hasil pengujian perangkat lunak membuktikan sistem ini mampu untuk memberikan informasi terkait suatu objek wisata serta beberapa fitur yang memanfaatkan *location based service.*

2.2. Konsep Dasar Program

Menurut Raharjo, dkk (2012:35) "Beberapa ciri dari *Object Oriented Programing* adalah abstraksi (*abstraction)*, pembungkusan (*encapsulation*), pewaris (*inheritance*), dan polimorfisme atau kebanyakanrupaan (*polymorphism*)."

Program merupakan kombinasi yang disusun atau ekspresi pernyataan sehingga dapat dirangkai menjadi satu kesatuan prosedur yang berupa urutan langkah-langkah dalam menyelesaikan masalah yang diimplementasikan dengan menggunakan bahasa pemograman, sehingga program bisa mengeksekusi. Bahasa pemograman merupakan bahasa yang bisa diterjemahkan menjadi kumpulan perintah-perintah dasar.Dalam penerjemahannya dapat dilakukan dengan program komputer yang disebut kompilator.

Didalam pembuatan program tentunya tidak bisa terlepas dari tahapan-tahapan yang harus dikerjakan secara terstruktur agar membantu penulis dalam menyelesaikan programnya dengan baik. Untuk lebih menjelasnya tahapan-tahapan perancangan program secara umum adalah sebagai berikut :

1. Definisi Masalah
   Dalam tahap ini penulis melakukan pengumpulan kebutuhan seluruh elemen seperti data-data yang digunakan.Dimana elemen-elemen itu dapat memanjang dalam pengembangan sistem informasi obyek wisata di Pemalang.
2. Analisis Kebutuhan

Kemudian ditahap menganalisa kebutuhan sistem dilakukan pengumpulan kebutuhan dengan fokus pada perangkat lunak, meliputi : informasi, fungsi masing-masing pada bagian sistem, kerja atau cara kerja, antar muka. Lalu menyediakan perangkat dan teknik yang dapat membantu penulis untuk menentukan kebutuhan melalui sistem yang telah berjalan pada sistem informasi obyek wisata dalam melakukan pengolahan data-datanya. Perangkat yang dimaksud adalah penggunaan Diagram Alir Data (DAD) untuk menyusun data *input*, proses dan *output*.

3. Desain Algoritma

   Menulis langkah-langkah didalam pemecahan yang ada dengan menggunakan simbol-simbol untuk menceritakan aktivitas data yang akan dikelola menjadi sebuah informasi.

4. Pengkodean
   Merupakan pengkodean dari algoritma yang dibuat, diterjemahkan kedalam bentuk *statement* yang sesuai dengan bahasa pemograman yang digunakan.
5. Melakukan Tes Program
   Dari proses logika yang sudah dibuat, akan diperiksa apakah program tersebut sudah benar sehingga bebas dari kesalahan atau harus diperbaiki kembali. Semua kesalahan yang terjadi diperbaiki agar program komputer bisa dijalankan dan memberikan hasil sesuai yang diharapkan.
6. Dokumentasi Program
   Pembuatan pendokumentasian program sebagai cadangan atau *backup* dimana proses ini penting untuk dilakukan, untuk usaha pengembangan program selanjutnya.
7. Pemeliharaan
   Pemeliharaan digunakan untuk menjabarkan aktivitas dari analisis sisitem pada saat perangkat telah dipergunakan oleh pemakai.

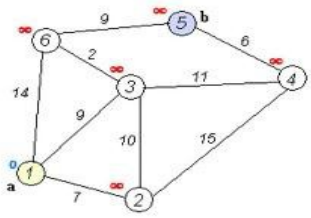
Untuk dapat membuat program komputer, harus mengenal dan menguasai beberapa bahasa komputer. Berbagai bahasa komputer sudah banyak diciptakan

untuk membantu manusia memprogram computer. Berdasarkan tingkatnya bahasa pemograman terdiri dari :

a. Bahasa Pemograman tingkat dasar antara lain : Bahasa *Assembley* dan Bahasa Mesin.
b. Bahasa Pemograman tingkat menengah antara lain : Bahasa C dan Bahasa *FORTH*.
c. Bahasa Pemograman tingkat atas antara lain : *FORTRAN*, Bahasa Pascal, COBOL dan lain-lain.

2.3. Algoritma *Djikstra*

Algoritma penyelesaian masalah yang digunakan adalah dengan mengunakan algoritma *Dijkstra y*ang di terapkan pada google maps.Algoritma *dijkstra* ini diterapkan pada sebuah aplikasi *location based service* berbasiskan platform Android yang memanfaatkan Google Map sebagai peta navigasi.Algoritma *Dijkstra* adalah sebuah algoritma *greedy* yang dipakai dalam memecahkan permasalahan jarak terpendek untuk sebuah *graf* berarah dengan bobot-bobot sisi (*edge)* yang bernilai tak negatif . Ide dasar algoritma *Dijkstra* sendiri ialah pencarian nilai *cost* yang terdekat dengan tujuan yang berfungsi pada sebuah graf berbobot, sehingga dapat membantu memberikan pilihan jalur. Pada Algoritma *Dijkstra*,*node* digunakan karena Algorima *Dijkstra* menggunakan *graph* berarah untuk penentuan rute lintasan terpendek. Algoritma ini bertujuan untuk menemukan jalur terpendek berdasarkan bobot terkecil dari satu titik ke titik lainnya.Misalkan titik mengambarkan gedung dan garis menggambarkan jalan, maka Algoritma *Dijkstra* melakukan kalkulasi terhadap semua kemungkinan bobot terkecil dari setiap titik.Pada Gambar 2 disajikan contoh *graf* dengan bobotnya dalam menentukan jalur menggunakan Algoritma *Dijkstra.*



Gambar II.**1** Contoh menemukan jalur menggunakan algoritma *Dijkstra*

Pertama-tama tentukan titik mana yang akan menjadi *node* awal, lalu beri bobot jarak pada *node* pertama ke *node* terdekat satu per satu, Algoritma *Dijkstra* akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap. Urutan logika dari Algoritma *Dijkstra* sebagai berikut.

1. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu *set* nilai 0 pada *node* awal dan nilai tak hingga terhadap *node* lain (belum terisi).

2. *Set* semua *node* belum terjamah dan *set node* awal sebagai *node* keberangkatan.

3. Dari *node* keberangkatan, pertimbangkan *node* tetangga yang belum terjamah dan hitung jaraknya dari titik keberangkatan. Sebagai contoh, jika titik keberangkatan A ke B memiliki bobot jarak 6 dan dari B ke *node* C berjarak 2, maka jarak ke C melewati B menjadi 6+2=8. Jika jarak ini lebih kecil darijarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.

4. Saat kita selesai mempertimbangkan setiap jarak terhadap *node* tetangga, tandai *node* yang telah terjamah sebagai *node* terjamah. *Node* terjamah tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.

5. *Set node* belum terjamah dengan jarak terkecil (dari *node* keberangkatan) sebagai *node* keberangkatan selanjutnya dan lanjutkan dengan kembali ke *step* 3.

2.4. Pengujian Aplikasi

Setelah program dibuat maka program tersebut perlu diuji dan diaplikasikan untuk membuktikan bahwa program yang telah dibuat telah berhasil menyesuakian sistem dengan persoalan. Program tersebut dapat diuji dengan caradua cara yaitu *Black box test* dan *white box testing* yang sekaligus bisa melakukan simulasi.

1. Metode Pengujian *Black Box*

Menurut Rizky (2011:264) menjelaskan bahwa "*Black box testing* adalah tipe testing yang memperlukan perngkat lunak yang tidak diketahui kinerja internalnya." Dengankata lain *Black box testing* merupakan sebuah metode pengujian perangkat lunak yang melakukan *test* fungsional dari aplikasi yang bertentangan dengan struktur *internal* atau struktur kerja.

2. Metode Pengujian *White Box*

Menurut Rizky (2011:261) menjelaskan bahwa "*White box* testing secara umum merupakan jenis testing yang lebih berkonsentrasi terhadap "isi" dari perangkat lunak itu sendiri." Pengujian *white box* juga bisa disebut pengujian struktur atau *structural testing,* jika ada suatu modul yang menghasilkan *output* yang tidak sesuai dengan proses yang dilakukan, maka variable, parameter dan baris-baris program yang terlibat pada unit tersebut akan diperiksa satu persatu dan setelah itu akan diperbaiki, kemudian akan di-*compile* ulang.

2.5. Peralatan Pendukung

1. Eclipse

Menurut Safaat (2015:4) mendefinisikan bahwa "*Eclipse* adalah IDE yang paling populer untuk pengembangan Android, karna memiliki Android *plug-in* yang tersedia untuk memfasilitasi pengembang android." Selain itu *Eclipse* juga mendapat dukungan dari Google untuk menjadi IDE pengembangan aplikasi android, ini terbukti dengan adanya penambahan *plugins* untuk *eclipse* untuk membuat project android dimana *source sofware* langsung dari situs resminya Google. Tetapi hal diatas tidak menutup kemungkinan untuk menggunakan IDE yang lain seperti *Netbeans* untuk melakukan pengembangan android.

Aplikasi android dapat dikembangkan pada sistem oprasi berikut:

a. Windows XP/Seven.
b. Mac OS X (Mac OS X 10.4.8 atau lebih baru.)
c. Linux.

2. Android *Sofware Development* Kit (SDK)

Menurut Safaat (2015:5) mendefinisikan bahwa "Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* android menggunakan bahasa pemerograman java " .Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem oprasi, *middleware* dan aplikasi kunci yang di *release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada platform android menggunakan bahasa pemerograma java. Sebagai *platform* aplikasi-netral, android memberi anda kesempatan untuk membuat aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan *Handphone/Smartphone*. Beberapa fitur-fitur Android yang paling penting adalah:

a. *Framework* Aplikasi yang mendukung penggantian komponen dan *reusable.*
b. Mesin *Virtual Dalvik* dioptimalkan untuk perangkat *mobile.*
c. *Integrated browser* berdasarkan *enggine open source Webkit*.

d. Grafis yang di optimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan sefesipikasi opengl ES 1,0 (*Opsional akselerasi hardware*).
e. SQLite untuk penyimpanan data.
f. Media Support yang mendukung audio, video, dan gambar (MPEG4, H.264, AAC, AMR, JPG, PNG, GIF), GSM Telephony (tergantung hardware)
g. Bluetooth, EDGE, 3G, dan WiFi (tergantung *hardware*).
h. Kamera, GPS, kompas, dan *accelerometer* (tergantung *hardware*).
i. Lingkungan Develpment yang lengkap dan kaya termasuk perangkat emulator, *tools* untuk *debugging*, profil dan kinerja memori, dan plugin untuk IDE Eclipse.

3. Android *Development Tools* (ADT)

Menurut Safaat (2015:5) mendefinisikan bahwa "*Android Development Tools* (ADT) adalah *plugin* yang didesain untuk IDE *Eclipse* yang memberikan kita kemudahan dalma mengembangkan aplikasi android dengan menggunakan IDE *Eclipse*." Dengan menggunakan ADT untuk *Eclipse* akan memudahkan kita dalam membuat aplikasi project android, membuat GUI aplikasi, dan menambahkan komponen-komponen menggunakan andrid SDK melalui *Eclipse*. Dengan ADT juga kita dapat melakukan pembuatan *package* android (.apk) yang digunakan untuk distribusi aplikasi yang kita rancang.

Mengembangkan aplikasi android dengan menggunakan ADT di *eclipse* sangat dianjurkan dan sangat mudah untuk memulai mengembangkan aplikasi android. Berikut adalah versi ADT untuk *Eclipse* yang sudah dirilis:

1. ADT 12.0.0 (July 2011)
2. ADT 11.0.0 (June 2011)
3. ADT 10.0.1 (March 2011)
4. ADT 10.0.0 (February 2011)
5. ADT 9.0.0 (January 2011)
6. ADT 8.0.1 (December 2010)
7. ADT 8.0.0 (December 2010)
8. ADT 0.9.9 (September 2010)
9. ADT 0.9.8 (September 2010)
10. ADT 0.9.7 (May 2010)

11. ADT 0.9.6 (March 2010)
12. ADT 0.9.5 (December 2009)
13. ADT 0.9.4 (October 2009)

Semakin tinggi *platform* android yang kita gunakan, dianjurkan menggunakan ADT yang lebih terbaru, karena biasanya menculnya platform baru akan diikuti oleh munculnya versi ADT yang terbaru.

4. Android *Virtual Device* (AVD)

Android Virtual Device adalah sebuah emulator Android, berfungsi sebagai emulator untuk mencoba aplikasi apakah berjalan dengan baik atau tidak sebelum dijalankan di *smartphone* dan Tablet Android. *Android Virtual Devices* (AVD) adalah konfigurasi dari emulator sehingga kita dapat menjalankan perangkat Android sesuai model yang dipilih, misal Android 1.5, 2.2 atau 2.3 dan versi Android selanjutnya.

5. XML

Menurut Kurniawati, dkk (2015:13) mendifinisikan bahwa "Xml *Layout* merupakan file yang memiliki elemen-elemen untuk menampilkan antara muka sebuah aplikasi android." Pada peroject Hello Android, memiliki sebuah file XML *Layout* yaitu *activity_main.xml* yang diletakan dibawah sub folder layout dari res. Tekan dua kali pada file *activity_main.xml*.

6. HIPO

Menurut Jogiyanto (2007:787) "*HIPO* (*Hierarchy plus inputoproces-Output*) merupakan metodologi yang dikembangkan dan didukung oleh IBM". Tetapi saat ini *HIPO* juga banyak di gunakan sebagai alat desain dan teknik dokumentasi dalam siklus pengembangan sistem. Fungsi utamanya adalah sebagai berikut:

a. Untuk menyediakan suatu struktur guna memahami fungsi fungsi dari *system* untuk lebih menekankan fungsi-fungsi yang harus di selesaikan oleh program yang digunakan untuk melaksanakan fungsi tersebut.
b. Untuk menyediakan penjelasan yang jelas dari input yang harus digunakan danoutput yang dihasilkan oleh masing-masing fungsi tiap-tiap tingkatan dari diagram-diagram *HIPO*.
c. Untuk menyediakan output yang tepat dan sesuai dengan kebutuhan pemakai.

Jenis-jenis Diagram dalam paket *HIPO* berisi tiga jenis diagram, yaitu :

1) Daftarisi Visual/ *Visual Tabel of Contents* (*VTOC*), yang terdiri dari satu diagram hirarki atau lebih.*Visual Tabel of Contents* menggambarkan seluruh program *HIPO* baik rinci maupun ringkasan yang tersetuktur. Pada diagram ini nama dan nomor dari program *HIPO* diidentifikasi. Struktur paket diagram dan hubungan fungsi juga diidentifikasi dalam bentuk hirarki.
2) Diagram ringkas/ *Overview Diagram* yaitu seri diagram fungsional. Masing-masing diagram dihubungkan dengan salah satu fungsi sistem. Diagram ringkasan menggambarkan fungsi dan referensi utama dari suatu sistem. Fungsi dan referensi ini di perlukan oleh program untuk memperluas fungsi sampai uraian yang terkecil. Diagram iniberisi input, proses dan output dari fungsi khusus.

Input pada diagram ini berisi item-item data yang dipakai oleh proses, sedangkan proses merupakan urutan langkah-langkah yang menjelaskan fungsi yang sedang dijalankan untuk menghasilkan suatu *output*. *Output* berisikan item-item data yang dihasilkan dan diubah oleh proses.

Anak panah pada diagram ringkasan dari input ke proses menunjukan hubungan antara item data di input dan langkah-langkah proses, sedangkan

anak panah dari proses ke *output* menunjukan hubungan tahap-tahap proses dan item data output.

3) Diagram rincian/detail diagram yaitu suatu seri dengan fungsional dan masing-masing diagram dihubungkan dengan sebuah sub-fungsi sistem. Diagram rinci merupakan diagram yang paling rendah dalam diagram yang terdapat dalam paket *HIPO*. Diagram rinci berisi unsur-unsur paket dasar.

7. Flowchart

Menurut Kadir (2012:12) "diagram alir (*flowchart*) adalah suatu standar untuk mengembangkan suatu peroses. Setiap langkah dalam algoritma dinyatakan dengan sebuah simbol dan aliran setiap langkah (dari suatu langkah ke langkah yang lain) dinyatakan dengan garis yang dilengkapi panah."

8. OOP (*Object Oriented Program*)

Menurut Sukamto dan Shalahudin (2014:100) Menyebutkan bahwa "Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya." Metode berorientasi objek juga meliputi rangkaian aktivitas analisis berorientasi objek, pemograman berorinteasi objek, perancangan berorientasi objek dan pengujian berorientasi objek banyak dipilih karena metode lama banyak menimbulkan masalah, seperti adanya kesulitan pada saat mentrasformasikan hasil dari satu tahapan pengembangan ketahapn pengembangan selanjutnya. Adapun keuntungan menggunakan metode berorientasi objek diantaranya sebagai berikut :

a. Meningkatkan produktivitas

Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang akan melibatkan objek tersebut.

b. Kemudahan pemeliharaan

Karena dengan model model, pola yang cendrung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

c. Kecepatan pengembangan

Karena sistem yang dibangun dengan baik dan benar pada saat analisa dan perangcangan akan mengakibatkannya berkurangnya kesalahan pada saat pengkodean.

d. Menigkatkan kualitas perngkat lunak

karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya kosisten pada saat pengembangannya, perangkat lunak yang akan dihasilkan bisa mampu memenuhi dari kebutuhan pemakai serta mempunyai sedikit kesalahan.

e. Adanya konsistensi

Karena sifat dari pewaris dan juga pengguna notasi yang sama pada saat analisa, perancangan maupun pengkodean.

Pada saat ini sudah banyak bahasa pemograman yang berorientasi objek. Banyak yang telah berfikir bahwa pemograman dengan berorientasi objek identik dengan bahasa java. Memang bahasa java merupakan bahasa yang paling konsisten dalam mengimplementasikan paradigm pemograman yang mendukung pemograman berorientasi objek tidak hanya bahasa java, berikut adalah beberapa konsep dasar yang harus dipahamitentang metologi berorientasi objek.

1) kelas (*class)*

Menurut Sukamto dan Shalahudin (2014:104) menyatakan bahwa " Kelas adalah kumpulan objek dengan karateristik yang sama". Sebuah kelas akan mempunya sifat *(atribute),* kelakuan *(method),* hubungan *(relationship)* dan arti.

2) Objek

Menurut Sukamto dan Shalahudin (2014:106) menyatakan bahwa "Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak". Objek merupakan suatu entitas yang mampu bisa menyimpan informasi (status) dan mempunyai oprasi (kelakuan) yang dapat diterapkan data dapat berpengaruh pada status objek.

3) Method *(method)*

Menurut Sukamto dan Shalahudin (2014:107) menyatakan bahwa "Operasi atau metode atau *method* pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural." Pada metodologi structural sebuah kelas boleh memiliki lebih dari satu metode atau operasi.Metode atau operasi yang bisa berfungsi untuk memanipulasi objek itu sendiri.Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

4) Atribut (*attribute*)

Menurut Sukamto dan Shalahudin (2014:108) menyatakan bahwa "Atribut dari sebuah kelas adalah variable global yang dimliki sebuah kelas."Atribut juga bisa berupa nilai atau element-element data yang

dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oelh sebuah objek, misalnya berat, jenis, nama, dan sebagainya

5) Abstraksi (*abstraction*)

Menurut Sukamto dan Shalahudin (2014:108) menyatakan bahwa " Prinsip untuk merespresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan".

6) Enkapsulasi (*encapsulation*)

Menurut Sukamto dan Shalahudin (2014:108) menyatakan bahwa "Pembungkus atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya".

7) Pewaris (*inheritance)*

Menurut Sukamto dan Shalahudin (2014:109) menjelaskan bahwa "Mekanisme yang memungkinkan suatu objek pewaris sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya".

8) Polimorfism

Menurut Sukamto dan Shalahudin (2014:110) menjelaskan bahwa "Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yanag sama sehingga menghemat baris program".

9. *UML (Unfield Modeling Language)*

Menurut Sukamto dan Shalahudin (2014:133) menjekaskan bahwa "UML adalah salah satu standar bahasa yang banyak digunakan didunia industry untuk

mendefinisikan *requirement,* membuat analisi dan desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek".

Pada perkembangan teknik pemograman berorientasi objek telah muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemograman ang berbasisi objek yaitu *Unifield Modeling Language* (UML).UML muncul karena danya kebutuhan dalam pemodelan visual untuk mengimplementasikan, menggambarkan, membangun, dan dikumentasi dari sistem perangkat lunak. Dengan kata lain UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

a. *Class* Diagram

Menurut Sukamto dan Shalahudin (2014:141) menjelaskan bahwa "Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas kelas yang akan dibuat untuk membangun sistem". Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi seusai dengan kebutuhan sistem, susunan struktru kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas sebagai berikut :

a) Kelas main

Kelas yang memiliki fungsi awal dieksekusi ketikan sistem dijalankan.

b) Kelas yang menangani tampilan sistem

Kelas yang mendefinisikan dan mengatur tampilan pemakai.

c) Kelas yang diambil dari pendefinisian use case

Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisan use case.

d) Kelas yang diambil dari pendefinisian data

Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan kebasis data

b. *Use Case* Diagram

Menurut Sukamto dan Shalahudin (2014:155) menjelaskan bahwa "*Use case* atau diagram *use case* merupakan permodelan untuk kelakuan (*behavior)* sisitem informasi yang akan dibuat".*Use case* di deskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat, *use case* digunakan untuk mengetahui fungsi apa saja yang berada dalam sebuah sistem infoormasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada use case denga nama didefinisikan sesimple mungkin dan dapat dipahami, ada dua hal utama pada use case yaitu pendefinisian atau apa saja yang disebut aktor dan *use case*.

1. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat alur sistem informasi yang akan dibuat itu sendiri jadi walaupun jadi simbol dari actor merupakan gambar orang, tapi actor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sisitem sebagai unit unit yang saling bertukar pesan antar unit atau aktor.

c. *Activity* Diagram

Menurut Sukamto dan Shalahuddin (2014:161) menjelaskan bahwa " 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 bukan apa yang dilakukan aktor jadi aktivitas yang dapat dilakukan sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal hal sebagai berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan peroses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/user interface dimana setiap aktivitas dianggap memiliki sbuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

d. *Sequence* Diagram

Menurut Sukamto dan Shalahudin (2014:165) menjelaskan bahwa "Diagram *sequence* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek". Oleh karena itu untuk menggambarkan diagram sekuen maka harus mengetahui objek-objek yang terlibat dalam *use case* beserta metode-metode yang dimiliki kelas yang diimplementasikan menjadi objek itu. Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat jugga semakin banyak.