BAB II
TINJAUAN PUSTAKA

2.1. Teori Umum

2.1.1. Pengertian Komputer

Siallagan (2009:1), "Komputer adalah sebagai sekumpulan alat elektronik yang saling bekerja sama, alat menerima data *(input),* mengolah data *(process),* memberikan informasi *(output),* dan terkoordinasi dibawah kontrol program yang tersimpan dalam memorinya."

Mulyono (2010:1), "Komputer adalah seperangkat alat elektronik yang terdiri atas peralatan input, alat yang mengolah input, dan peralatan output yang memberikan informasi, serta bekerja secara otomatis."

Wahyudi (2012:3), "Komputer adalah peralatan (*device*) yang menerima data (*input*) dan menyimpan (*storage*) kemudian diproses (*process*) untuk menghasilkan data dalam bentuk lain (*output*)."

Dari beberapa definisi diatas penulis menyimpulkan bahwa komputer adalah alat elektronik yang dapat mengelola data menjadi informasi yang berguna.

2.1.2. Pengertian Data

Menurut McLeod dalam Yakub (2012:5), "Data adalah kenyataan yang menggambarkan adanya suatu kejadian (*event*), data terdiri dari fakta (*fact*) dan angka yang secara relatif tidak berarti bagi pemakai."

Sutabri (2012:25), "Data merupakan bentuk mentah yang belum dapat bercerita banyak sehingga perlu diolah lebih lanjut."

Asropudin (2013:22), "Data adalah kumpulan dari angka-angka maupun karakter-karakter yang tidak memiliki arti."

Dari beberapa definisi diatas penulis menyimpulkan bahwa data adalah sekumpulan fakta yang harus diolah terlebih dahulu agar menghasilkan suatu informasi.

2.1.3. Pengertian Basis Data *(Database)*

Sutanta (2011:35), "Basis data merupakan sekumpulan dari bermacam-macam tipe *record* yang memiliki hubungan antar-*record* dan rincian data terhadap obyek tertentu."

Badiyanto (2013:57), "*Database* bisa dikatan sebagai suatu kumpulan dari data yang tersimpan dalam tabel dan diatur atau diorganisasikan sehingga data tersebut bisa diambil atau dicari dengan mudah dan efisien."

Indrajani (2014:70), "Sebuah basis data adalah sebuah kumpulan data yang saling berhubungan secara logis, dan merupakan sebuah penjelasan dari data tersebut, yang didesain untuk menemukan data yang dibutuhkan oleh sebuah organisasi."

Dalam terminologi database relasional, dikenal istilah seperti:

a. Tabel

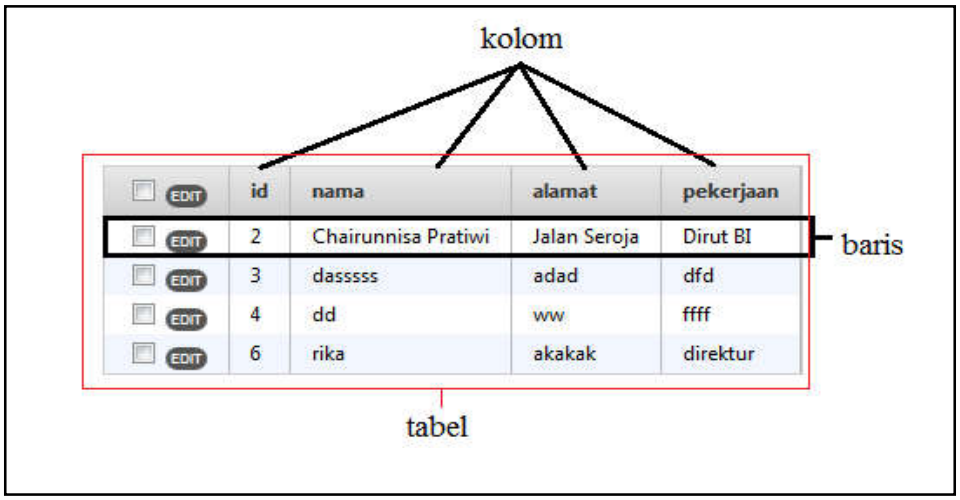
   Tabel menyatakan bentuk berdimensi dua yang mewakili suatu kelompok data yang sejenis.

b. Kolom (*field*)

   Kolom atau *field* adalah data yang berurut-urut berisi informasi secara vertikal.

c. Baris (*record*)

   Baris atau *record* adalah data yang tersusun secara horizontal.



Gambar 2.1. Tabel Kolom dan Baris pada *Database*

Dari beberapa definisi diatas penulis menyimpulkan bahwa basis data (*database*) adalah tempat yang digunakan untuk menyimpan data-data agar lebih terorganisir.

2.2. Teori Judul

2.2.1. Pengertian Aplikasi

Sutabri (2012:147), "Aplikasi adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya".

Sujatmiko (2012:23), "Aplikasi merupakan program komputer yang dibuat oleh suatu perusahaan komputer untuk membantu manusia dalam mengerjakan tugas-tugas tertentu, misalnya *Microsoft Word, Microsoft Excel"*.

2.2.2. Pengertian Reservasi *(Reservation)*

(1987:14), "Reservasi yaitu pemesanan fasilitas yang diantaranya akomodasi, meal, seat pada pertunjukan, pesawat terbang, kereta api, bus, hiburan, night club, discoutegue dan sebagainya".

2.2.3. Pengertian Perjalanan

Perjalanan adalah kegiatan manusia untuk mengunjungi suatu tempat dengan berbagai tujuan.

Perjalanan dapat didefinisikan sebagai kegiatan berpindah dari satu tempat ke tempat lain dengan berbagai tujuan.

(http://novianto-anto-fisip.web.unair.ac.id/artikel_detail-70003-Umum-Pengantar%20Pariwisata.html Diakses pada tanggal 24 juni 2016 pukul 18.40 wib)

2.2.4. Pengertian Wisata

Wisata adalah bepergian secara bersama-sama dengan tujuan untuk bersenang-senang, menambah pengetahuan, dan lain-lain. Selain itu juga dapat diartikan sebagai bertamasya atau piknik.

http://anekatempatwisata.com/pengertian-wisata-secara-umum/ Diakses pada tanggal 24 juni 2016 pukul 18.40 wib)

Wisata adalah perjalanan yang dilakukan oleh seseorang atau kelompok orang dengan mengunjungi tempat tertentu untuk tujuan rekreasi, pengembangan pribadi, atau mempelajari daya tarik wisata yang dikunjunginya dalam jangka waktu sementara.

Kegiatan perjalanan atau sebagian dari kegiatan tersebut yang dilakukan secara sukarela serta bersifat sementara untuk menikmati obyek dan daya tarik wisata.

http://novianto-anto-fisip.web.unair.ac.id/artikel_detail-70003-Umum-Pengantar%20Pariwisata.html Diakses pada tanggal 24 juni 2016 pukul 18.40 wib)

2.2.5. Pengertian Android

Irawan (2012:2), "Android merupakan sebuah sistem operasi yang berbasis Linux untuk perangkat portable seperti *smartphone* dan komputer tablet."

Juhara (2016:1), "Android adalah sistem operasi berbasis Linux yang dimodifikasi untuk perangkat bergerak (*mobile devices*) yang terdiri dari sistem operasi , *middleware*, dan aplikasi-aplikasi utama.

2.2.6. Fitur-fitur Android

Fitur-fitur yang tersedia di android adalah :

a. Kerangka aplikasi : itu memungkinkan penggunaan dan penghapusan komponen yang tersedia.
b. Dalvik mesin virtual : mesin virtual dioptimalkan untuk perangkat telepon seluler.
c. Grafik : grafik di 2D dan grafis 3D berdasarkan pustaka OpenGL
d. SQLite : untuk penyimpanan data.
e. Mendukung media : audio, videom dan berbagai format gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
f. GSMm *Bluetooth*, EDGE, 3G, 4G dan WiFi
g. Kamera, *Global Positioning System* (GPS), kompas, NFC dan *accelerometer*.

2.2.7. Generasi Android

Ponsel pertama yang memakai sistem Operasi Android adalah HTC Dream yang di rilis pada tanggal 22 Oktober 2008 dan pada awal tahun 2009

mulailah para pengembang ponsel menggunakan OS android ini dan di perkirakan setidaknya 18 ponsel bersistem OS Android rilis di awal tahun 2009.

Masruri (2015:4), perkembangan android adalah sebagai berikut :

1. **Android 1.1 Bender**
   a. Pertama kali dirilis pada 9 Februari 2009. Pada awalnya Android ini akan diberi nama "Bender" akan tetapi karena alasan melanggar trademark, nama "Bender" tidak jadi disematkan pada versi Android ini. Awalnya versi OS Android ini dirilis untuk perangkat T-Mobile G1 saja. Versi ini merupakan *update* untuk memperbaiki beberapa *bugs*, mengganti API dan menambahkan beberapa fitur.
2. **Android 1.5 Cupcake**
   a. Pertama kali dirilis pada 30 April 2009. Nah, mulai versi Android ini penamaan menggunakan nama makan pencuci mulut (*dessert*) mulai digunakan, karena ini merupakan versi yang ketiga maka penamaan diawali dengan huruf "C" dan jadilah "*Cupcake*" menjadi nama resmi dari versi OS Android ketiga ini. OS ini berbasiskan pada *kernel* Linux 2.6.27 dan menambahkan beberapa update serta UI baru dari versi Android sebelumnya. Mulai terdapat "*widget*" yang dapat dibesar kecilkan. Kemudian ditambah kemampuan untuk meng-*upload* video dan gambar ke Youtube dan Picasa.
3. **Android 1.6 Donut**
   a. Dirilis pertama kali pada 15 September 2009. Terdapat peningkatan pada fitur pencarian dan UI yang lebih *user friendly*. Pada versi ini juga sudah mendukung teknologi CDMA/EVDO, 802.1x, VPNs. Kemudian *support* layar dengan resolusi WVGA. Berikut penampakan Android v1.6 Donut.
4. **Android 2.0/2.1 Éclair**
   a. Dirilis pertama kali pada 9 Desember 2009. Terjadi penambahan fitur untuk pengoptimalan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML5, daftar kontak yang baru, dukungan flash untuk kamera 3,2 MP, digital Zoom, dan Bluetooth 2.1. Beberapa versi updatenya antara Android v.2.0 kemudian v2.0.2 dan terakhir v.2.1.

5. Android 2.2 Froyo (Froze Yoghurt)

a. Dirilis pertamakali pada 20 Mei 2010 pada *smartphone* Google Nexus One. Pada versi ini sudah *support* terhadap Adobe Flash Player 10.1. Peningkatan pada kecepatan membuka dan menutup aplikasi, serta penggunaan SD Card sebagai tempat penyimpanan aplikasi. Ketika Android Froyo hadir mulai muncul banyak diskusi yang membahas mengenai persaingan antara Android dengan iOS yang akan semakin ketat di masa yang akan datang. Beberapa versi *update* yang dirilis antara lain Android v.2.2.1 hingga v.2.2.3.

6. Android 2.3 Gingerbread

a. Pertama kali diperkenalkan pada 6 Desember 2010. Terjadi banyak peningkatan pada versi Android yang satu ini dibandingkan dengan versi sebelumnya. Dirancang untuk memaksimalakan kemampuan aplikasi dan game. Serta mulai digunakannya *Near Field Communication* (NFC). Perbaikan terhadap dukungan layar resolusi WXGA dan diatasnya. Beberapa versi *update* yang dirilis antara lain v.2.3.3 hingga v.2.3.7. Sampai saat ini Android Gingerbread merupakan versi Android yang memiliki pengguna terbanyak dibandingkan dengan seri Android lainnya, yaitu mencapai 65% dari seluruh versi Android yang dirilis.

7. Android 3.0/3.1 Honeycomb

a. Pertama kali diperkenalkan pada 22 Februari 2011 dan Motorola Xoom adalah yang pertama kali menggunakannya. Android versi ini merupakan OS yang didesain khusus untuk pengoptimalan pengunaan pada tablet PC.

8. Android 4.0 ICS (Ice Cream Sandwidch)

a. Sampai tulisan ini ditulis ICS merupakan versi Android yang paling anyar. Pertama kali dirilis pada 19 Oktober 2011. *Smartphone* yang pertama kali mengunakan OS Android ini adalah Samsung Galaxy Nexus. Secara teori semua perangkat seluler yang menggunakan versi Android sebelumnya, Gingerbread, dapat di-*update* ke Android Ice Cream Sandwich.

9. Android versi 4.1 (Jelly Bean)

a. Android Jelly Bean yaang diluncurkan pada acara Google I/O lalu membawa sejumlah keunggulan dan fitur baru. Penambahan baru diantaranya

meningkatkan input *keyboard*, desain baru fitur pencarian, UI yang baru dan pencarian melalui *Voice Search* yang lebih cepat. Tak ketinggalan *Google Now* juga menjadi bagian yang diperbarui. *Google Now* memberikan informasi yang tepat pada waktu yang tepat pula. Salah satu kemampuannya adalah dapat mengetahui informasi cuaca, lalu-lintas, ataupun hasil pertandingan olahraga. Sistem operasi Android Jelly Bean 4.1 muncul pertama kali dalam produk tablet Asus, yakni Google Nexus 7.

10. Android versi 4.4 (KitKat)

a. Sebelum dirilis resmi, para pengamat gadget memprediksi bahwa untuk versi lanjutan dari Jelly Bean akan diberi nama "Key Lime Pie", tapi ternyata rumor tersebut salah kaprah. Penamaan untuk versi ini cukup mencengangkan karena mengambil nama produk coklat yang memang sudah terkenal sebelumnya atau tergolong komersial. Fitur yang diberikan Android KitKat yaitu Multitasking yang lebih cepat, Mendukung aksesbilitas yang lebih baik, Memudahkan akses file, Memudahkan mencetak dokumen, Peningkatan pengalaman membaca dengan eBook, Mendukung dua sensor baru, Peningkatan tampilan "art", Smaller Caller ID, Kontak Prioritas, "OK Google" dan Pemusatan SMS serta MMS.

11. Android Versi 5.0 (Lollipop)

a. Android Lollipop ini baru saja dirilis pada tanggal 15 Oktober 2014, Meskipun pada saat itu Lollipop baru dalam masa percobaan akan tetapi komentar yang masuk terbilang bagus. Fitur Android Lollipop yaitu Desain, Notifikasi, Hemat Baterai, Keamanan, Device Sharing, New Quick Settings, Android Tv dan Tambahan 15 bahasa baru.

2.2.8. Pengertian Aplikasi Reservasi Paket Perjalanan Wisata Berbasis Android pada CV. Nirwana Sembilan Benua *Tour & Travel* Palembang.

Aplikasi Reservasi Paket Perjalanan Wisata Berbasis Android pada CV. Nirwana Sembilan Benua *Tour & Travel* Palembang adalah Suatu Program yang dirancang untuk mempermudah pelanggan dalam melakukan pemesanan atau

reservasi paket perjalanan wisata yang dapat diakses dengan menggunakan *gadget* atau lebih dikenal dengan *Smartphone* yang terhubung dengan jaringan internet (*online*) agar memudahkan pelanggan dalam melakukan reservasi dimanapun dan kapanpun tanpa harus datang ke perusahaan.

2.3. Teori Khusus

2.3.1. Pemograman Berorientasi Objek

2.3.1.1. Pengertian Pemograman Berorientasi Objek

Sukamto dan Shalahuddin (2013:100), "Berorientasi Objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya."

Dari definisi diatas dapat disimpulkan bahwa pemrograman berorientasi objek adalah program komputer dari berbagai objek yang melakukan suatu tindakan terhadap masing-masing objek.

2.3.1.2. Ciri Pemrograman Berorientasi Objek

Siallagan (2009:149), Ciri-ciri atau karakteristik pemograman berorientasi objek, antara lain:

a. Abstraksi (*Abstraction*)

   Abstraksi adalah pengabstrakan atau melakukan seleksi terhadap aspek-aspek tertentu suatu masalah. Abstraksi digunakan untuk penyembunyian kerumitan dari suatu proses. Sebagai contoh, dalam membuat suatu sistem, ada tombol-tombol yang dapat digunakan. Operator atau pengguna tidak perlu berpikir tentang pembuatan tombol tersebut, tetapi yang penting mereka dapat menggunakannya.

b. Pembungkusan (*Encapsulation*)

   Pembungkusan sering pula disebut pengkapsulan. Artinya, proses membuat paket (memaketkan) data objek bersama dengan metode-metodenya. Berdasarkan kode program, proses memisahkan aspek-aspek objek dilakukan dengan pembungkusan. Proses pembungkusan itu sendiri merupakan cara atau mekanisme untuk melakukan abstraksi.

c. Pewarisan (*Inheritance*)

Pewarisan adalah memberikan atau mewariskan sesuatu kepada keturunan berikutnya. Misalnya, seorang anak pasti akan mewarisi beberapa sifat atau perilaku yang dimiliki oleh ibu/bapaknya. Dalam konteks ini, suatu kelas dalam program dapat diturunkan menjadi kelas-kelas baru lainnya yang akan mewarisi beberapa sifat atau perilaku dari kelas induknya.

d. Polimorfisme (*Polymorphism*)

Polimorfisme adalah suatu kejadian ketika objek dapat mengungkap banyak hal melalui satu cara yang sama.

2.3.2. Metodologi *RUP* (*Rational Unified Process*)

Sukamto dan Shalahudin (2014:30) "*Rational Unified Process (RUP)* merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai *best practises* yang terdapat dalam industri pengembangan perangkat lunak. Ciri khas metode ini adalah menggunakan *use-case driven* dan pendekatan iteratif untuk siklus pengembangan perangkat lunak. Gambar dibawah ini menunjukkan secara keseluruhan arsitektur yang dimiliki *RUP*."

RUP menggunakan konsep *object oriented*, dengan aktifitas yang berfokus pada pengembangan model dengan menggunakan *Unified Model Language* (*UML*).

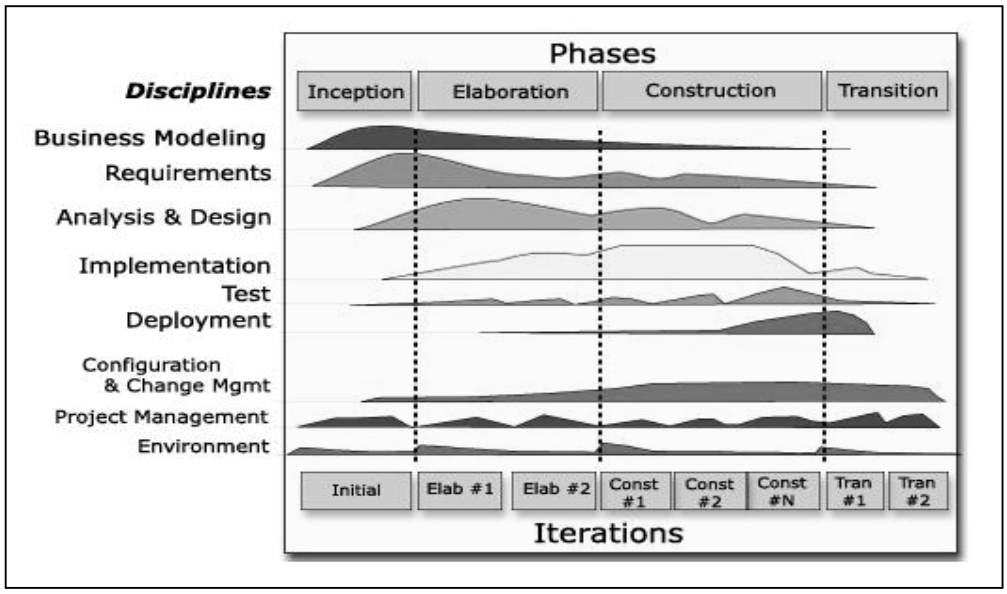
Melalui gambar dibawah dapat dilihat bahwa *RUP* memiliki, yaitu:

a. Dimensi Pertama

Digambarkan secara horizontal. Dimensi ini mewakili aspek-aspek dinamis dari pengembangan perangkat lunak. Aspek ini dijabarkan dalam tahapan pengembangan atau fase. Setiap fase akan memiliki suatu *major milestone* yang menandakan akhir dari awal dari *phase* selanjutnya. Setiap *phase* dapat berdiri dari satu beberapa iterasi. Dimensi ini terdiri atas *Inception, Elaborationn, Construction*, dan *Transition.*

b. Dimensi Kedua

Digambarkan secara vertikal. Dimensi ini mewakili aspek-aspek statis dari proses pengembangan perangkat lunak yang dikelompokkan ke dalam beberapa disiplin. Proses pengembangan perangkat lunak yang dijelaskan kedalam beberapa disiplin terdiri dari empat elemen penting, yakni *who is doing, what, how,* dan *when.* Dimensi ini terdiri atas: *Bussines Modeling, Requirement, Analysis and Design, Implementation, Test, Deployment, Configuration, dan Change Management, Project Management,Environment.*



Gambar 2.2 *Arsitektur Rational Unified Process*

2.3.2.1.Penerapan Tahap Metodologi Pengembangan Perangkat Lunak dengan *RUP*

Sukamto dan Shalahudin (2011:32) dalam *Rational Unified Process* terdapat empat tahap pengembangan perangkat lunak yaitu:

a. Inception

Pada tahap ini pengembangan mendefinisikan batasan kegiatan, melakukan analisis kebutuhan *user*, dan melakukan perancangan awal perangkat lunak

(perancangan arsitektural dan *use case*). Pada akhir fase ini prototipe perangkat lunak versi *Alpha* harus sudah dirilis.

b. *Elaboration*

Pada tahap ini dilakukan perancangan perangkat lunak mulai dari menspesifikasikan fitur perangkat lunak hingga perilisan prototipe versi *Betha* dari perangkat lunak.

c. *Construction*

Pengimplementasian rancangan perangkat lunak yang telah dibuat dilakukan pada tahap ini. Pada akhir tahap ini, perangkat lunak versi akhir yang sudah disetujui administrator dirilis beserta dokumentasi perangkat lunak.

d. *Transition*

Instanlasi, deployment dan sosialisasi perangkat lunak dilakukan pada tahap ini.

2.3.2.2. Aliran Kerja Utama *RUP*

Sukamto dan Shalahudin (2011:32) adapun aliran kerja utama pada Metodologi *RUP* adalah sebagai berikut:

a. Pemodelan Bisnis (*Bussines Modeling*)

Mendeskripsikan struktur dan proses-proses bisnis organisasi.

b. Kebutuhan (*Requirement*)

Mendefinisikan kebutuhan perangkat lunak dengan menggunakan metode *use case.*

c. Analisis dan Perancangan (*Analysis and Design*)

Mendeskripsikan berbagai arsitektur perangkat lunak dari berbagai sudut pandang.

d. Implementasi (*Implementation)*

Menuliskan kode-kode program, menguji, dan mengintegrasikan unit-unit programnya.

e. Pengujian (*Test*)

Mendeskripsikan kasus uji, prosedur, dan alat ukur pengujian.

f. *Deployment*

Menangani konfigurasi sistem yang akan diserahkan.

2.3.2.3. Aliran Kerja Pendukung *RUP*

Sukamto dan Shalahudin (2011:31) adapun aliran kerja pendukung *RUP* adalah sebagai berikut:

a. Manajemen konfigurasi dan perubahan (*configuration and change management*) mengendalikan perubahan dan memelihara artifak-artifak proyek.
b. Manajemen proyek (*Project Management*)

   Mendeskripsikan berbagai stategi pekerjaan dengan proses yang berulang.
c. Lingkungan (*Environment*)

   Menangani infrastuktur yang dibutuhkan untuk mengembangkan sistem.

2.3.3. Metode Pengujian Perangkat Lunak

2.3.3.1. Pengertian Metode Pengujian

Shihab (2011), Metode pengujian adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan.

Pengujian perangkat lunak perlu dilakukan untuk mengevaluasi baik secara manual maupun otomatis untuk menguji apakah perangkat lunak sudah memenuhi persyaratan atau belum, dan untuk menentukan perbedaan antara hasil yang diharapkan dengan hasil sebenarnya.

2.3.3.2. Metode Pengujian

Perangkat lunak dapat diuji dengan dua cara, yaitu:

1. Pengujian dengan menggunakan data uji untuk menguji semua elemen program (data internal, loop, keputusan dan jalur). Data uji dibangkitkan dengan mengetahui struktur internal (kode sumber) dari perangkat lunak.

2. Pengujian dilakukan dengan mengeksekusi data uji dan mengecek apakah fungsional perangkat lunak bekerja dengan baik. Data uji dibangkitkan dari spesifikasi perangkat lunak.

2.3.3.3. Metode *Black Box Testing*

Pressman (2010:597), "*Pengujian kotak hitam*, juga disebut *pengujian perilaku,* berfokus pada persyaratan fungsional perangkat lunak. Artinya, teknik pengujian kotak hitam memungkinkan untuk membuat beberapa kumpulan kondisi masukan yang sepenuhnya akan melakukan semua kebutuhan fungsional untuk program. Pengujian kotak hitam berupaya untuk menemukan kesalahan dalam kategori berikut: fungsi yang salah atau hilang, kesalahan antarmuka, kesalahan dalam struktur data atau akses basis data eksternal, kesalahan perilaku atau kinerja dan kesalahan inisialisasi dan penghentian.

Shihab (2011), "*Black Box Testing* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program."

Shihab (2011), mengemukakan ciri-ciri *black box testing*, yaitu:

1. *Black box testing* berfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software*.
2. *Black box testing* bukan teknik alternatif daripada *white box testing*. Lebih dari pada itu, ia merupakan pendekatan pelengkap dalam mencakup error dengan kelas yang berbeda dari metode *white box testing*.

Black box testing melakukan pengujian tanpa pengetahuan detil struktur internal dari sistem atau komponen yang dites. juga disebut sebagai *behavioral testing, specification-based testing, input/output testing* atau *functional testing.*

2.3.4. *Unified Modeling Language (UML)*

Sukamto dan Shalahuddin (2013:133), *UML* (*Undefied Modelling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia

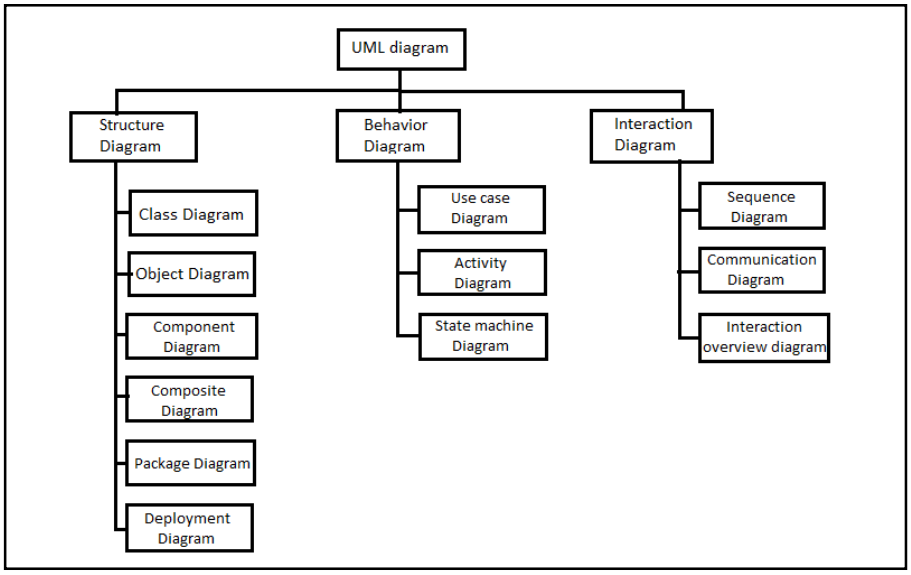
industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

UML menyediakan bahasa pemodelan visual yaitu proses penggambaran informasi-informasi secara grafis dengan notasi-notasi baku yang telah disepakati sebelumnya. Dengan menggunakan pemodelan *UML* ini, pengembang dapat melakukan:

a. Tinjauan umum bagaimana arsitektur sistem secara keseluruhan.
b. Penelaahan bagaimana objek-objek dalam sistem saling mengirimkan pesan (*message*) dan saling bekerjasama satu sama lain.
c. Menguji apakah sistem/perangkat lunak sudah berfungsi seperti yang seharusnya.
d. Dokumentasi sistem/perangkat lunak yntuk keperluan-keperuluan tertentu di masa yang akan datang.

2.3.4.1. Macam-macam Diagram *UML*

Pada *UML* terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar dibawah ini:



Gambar 2.3 *Macam-macam Diagram UML*

Berikut ini penjelasan singkat dari pembagian kategori tersebut.

a. *Structure Diagram*

Yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

b. *Behavior Diagram*

Yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

c. *Interaction Diagram*

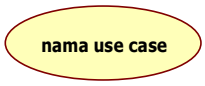
Yaitu kumpulan diagram yang digunakan untuk menggambarkan interkasi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

2.3.5. *Use Case Diagram*

Munawar (2005:62), "*Use case* adalah deskripsi fungsi dari sebuah system dari perspektif pengguna."

Sukamto dan Shalahuddin (2013:155), *Use case* atau diagram *use case* merupakan pemodelan untuk kelakukan (*behavior)* sistem informasi yang akan dibuat.

Tabel 2.1 Simbol-simbol dalam *Use case*

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Lanjutan Tabel 2.1 Simbol-simbol dalam *Use Case Diagram*

Aktor/Actor  	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
Asosiasi / association 	Komunitas antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor.
Ekstensi / extend 	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tamhanan itu; biasanya use case yang menjadi extend-nya merupakan jenis yang sama dengan use case yang menjadi induknya.
Generalisasi / *Generalization* 	Hubungan generalisasi dan spesifikasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

Lanjutan Tabel 2.1 Simbol-simbol dalam *Use Case Diagram*

Include <<include>> 	Relasi use case tambahan ke sebuah *use case* di mana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini; *include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan.

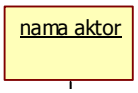
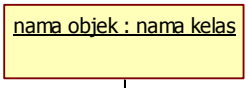
(**Sumber**: Sukamto dan Shalahuddin,2013:155)

2.3.6. *Sequence Diagram*

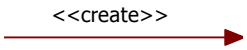
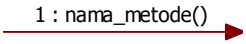
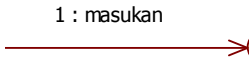
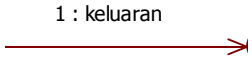
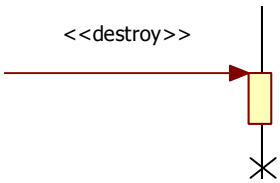
Munawar (2005:87), "*Sequence diagram* digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh obyek dam message (pesan) yang diletakkan diantara obyek-obyek ini di dalam use case."

Sukamto dan Shalahuddin (2013:165), "Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek."

Tabel 2.2 Simbol-simbol dalam *Sequence Diagram*

Simbol	Deskripsi
Aktor  atau  tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
Garis hidup / *lifeline* 	menyatakan kehidupan suatu objek.
Objek  	Menyatakan objek yang berinteraksi pesan.
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya.

Lanjutan Tabel 2.2 Simbol-simbol dalam *Sequence Diagram*

Pesan tipe create 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe call 	Menyatakan suatu objek memanggil operasi/ metode yang ada pada objek lain atau dirinya sendiri, arah panah mengarah pada objek yang memiliki operasi/ metode, karena ini memanggil operasi/ metode maka operasi/ metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
Pesan tipe send 	Menyatakan bahwa suatu objek mengirimkan data/ masukan/ informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
Pesan tipe return 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
Pesan tipe destroy 	Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaliknya jika ada create maka ada destroy.

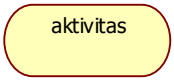
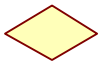
(**Sumber**: Sukamto dan Shalahuddin,2013:165)

2.3.7. *Acitivity Diagram*

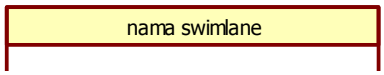
Munawar (2005:109), "*Activity Diagram* adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus."

Sukamto dan Shalahuddin (2013:161), "Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak."

Tabel 2.3 Simbol-simbol dalam *Activity Diagram*

Simbol	Deskripsi
Status awal 	Status awal aktivtas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
Penggabungan / *join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

Lanjutan Tabel 2.3 Simbol-simbol dalam *Activity Diagram*

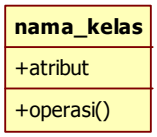
Swimlane  atau  	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

(**Sumber**: Sukamto dan Shalahuddin,2013:162)

2.3.8. *Class Diagram*

Sukamto dan Shalahuddin (2013:141), diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Tabel 2.4 Simbol-Simbol *Class Diagram*

Simbol	Deskripsi
Kelas  	Kelas pada struktur sistem.

Lanjutan Tabel 2.4 Simbol-Simbol *Class Diagram*

Antar muka/ *interface*  	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
Asosiasi / *association* 	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah / *directed association* 	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelass yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
Generalisasi 	Relasi antarakelas dengan makna generalisasi-spesialisasi (umum khusus).
Kebergantungan / *dependency* 	Relasi antarkelas dengan makna kebergantungan antarkelas.
Agregasi / *aggregation* 	Relasi antarkelas dengan makna semua-bagian (whole-part).

(**Sumber**: Sukamto dan Shalahuddin,2013:146)

2.4. Teori Program

2.4.1. Pemograman Java

2.4.1.1. Pengertian Pemrograman *Java*

Sukamto dan Shalahuddin (2013:103), "*Java* adalah bahasa pemrograman objek murni karena semua kode programnya dibungkus dalam kelas."

Asropudin (2013:52), "*Java* adalah bahasa pemrograman untuk menciptakan isi yang aktif dalam halaman *web*, juga dapat dijalankan dalam semua komputer."

2.4.1.2. Pengelompokan Tipe Data dalam *Java*

Sutanta (2005:422), Bahasa *Java* mengenal tipe data yang mirip dengan bahasa C++. Tipe ukuran memori yang dibutuhkan, dan batasan nilai data dalam java adalah sebagai berikut:

Tabel 2.5 *Tipe Data Dalam Java*

No.	Tipe data		Ukuran Memori	Batasan Nilai
1	*Integer*	*Int*	4 *byte*	-2.147.486.648 s/d 2.147.486.6
		Short	2 *byte*	-32.768 s/d 32.767
		Long	8 *byte*	-9.223.372.036.854.775.808L s/d 9.223.372.036.854.775.807L
		Byte	1 *byte*	-128 s/d 127
2	*floating point*	*Float*	4 *byte*	± 3.40282347E+38F (7 digit signifikan)
		double	8 *byte*	± 1.79769313486231570E+308 (15 digit signifikan)
3	karakter dan *string*	*Char*	1 karakter	Sebuah objek string dan manipulasinya
		String	karakter	

2.4.1.3. Mendeklarasi Variabel

Berikut ini bentuk umum cara mendeklarasi variabel pada bahasa *java*:

Tipe namaVariabel;

Tipe namaVariabel;

Contoh:

int lebar,tinggi;

float hasil;

2.4.1.4. Operator dalam *Java*

a. Operator Aritmatika

Operator aritmatika adalah operator-operator yang digunakan untuk mengoprasikan perhitungan (aritmatika). Bahasa pemograman *java* menyediakan operator-operator aritmatika untuk memanipulasi variabel data. Menurut Siallagan (2009:50).

Tabel 2.6 *Operator Aritmatika*

Operator	Keterangan
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Modulus (sisa bagi)

b. Operator Relasional

Operator relasional adalah operator hubungan (relasi) yang membandingkan kedua nilai *operand* dan hasilnya berupa nilai *boolean*, yaitu benar (*true*) atau salah (*false*). Menurut Siallagan (2009:65).

Tabel 2.7 *Operator Relasional*

Operator	Keterangan
==	Sama dengan (membandingkan bukan penugasan)
!=	Tidak sama dengan
>	Lebih besar
<	Lebih kecil
>=	Lebih besar sama dengan
<=	Lebih kecil sama dengan

c. Operator Logiaka/*Boolean*

Operator logika adalah operator yang digunakan terhadap *operand* bertipe *Boolean* yang hasilnya benar (*true*) atau salah (*false*). Menurut Siallagan (2009:51).

Tabel 2.8 *Operator Logika*

Operator	Keterangan
&	Logika AND
|	Logika OR
^	Logika XOR
!	Logika NOT

2.4.1.5. Java Sebagai Suatu *Platform*

Platform adalah suatu lingkungan *hardware* atau *software* yang suatu program dapat berjalan di dalamnya. Contoh *platform* adalah Microsoft Windows, Solaris OS, Linux, dan Mac OS. Suatu *platform* biasanya hasil kombinasi dari sistem operasi dan seperangkat *hardware*. Terminologi pemrograman *Java* bukan hanya merujuk pada suatu bahasa pemrogramman, melainkan juga sebagai suatu *platform*.

Tapi *platform Java* hanya terdiri dari komponen *software* (tanpa komponen *hardware*) dalam hal ini adalah untuk menjalankan *hardware. Platform Java*

1. Sebagai *Java Virtual Machine* (JVM)

   adalah mesin *virtual* yang menerjemahkan dan mengkomunikasikan *bytecode-bytecode java* ke dalam bahasa mesin. *Bytecode java* adalah adalah file hasil kompilasi kode *java* (ekstensi filenya adalah .class). Jika sutu program java bernama ProgramA.java dikompilasi maka hasilnya adalah Program A.class (inilah *bytecode java*). Sebenernya paradigma pemrogramman dengan cara mengkomunikasikan /menginterpretasikan kode (dalam java adalah *bytecode*) menurut para ahli kurang bagus dari sisi *performance* (kecepatan). Namun JVM mencoba mengatasi masalah ini dengan menerapkan teknik *just in time* (JIT) *compilation* yaitu *java bytecode* langsung dikompilasi menjadi bahasa mesin untuk kode-kode program yang dijalankan secara berulang-ulang.

2. Java Sebagai *Application Programming Interface* (API)

   API merupakan sekumpulan komponen *software* (kelas-kelas dan *interface-interface java*) siap pakai yang memiliki berbagai kegunaan dan kemampuan yang berbeda-beda. Sekumpulan kelas-kelas dan *interface-interface* yang saling berkaitan diorganisasikan dalam suatu daftar pustaka/ *library*. *Library* ini dikenal dengan sebutan *package* (paket). Beberapa fitur yang ditawarkan Java API antara Iain sebagai berikut :

   1. *Applet*
   2. *Java Networking*

3. *Java Database Connectivity* (JDBC)

   Java Database Connectivity (JDBC) adalah sebuah *Application Programming Interface* (API) pendukung bahasa pemrograman Java yang mendefinisikan bagaimana sebuah klien dapat mengakses sebuah database. JDBC menyediakan metode-metode untuk *query* dan *update* data dalam database. Java SE menyertakan JDBC API bersamaan dengan implementasi ODBC (*Open Database Connectivity* merupakan sebuah standar terbuka untuk konektivitas antar mesin basis data) untuk memudahkan koneksi ke database apa saja. JDBC disertakan dalam bentuk *driver* yang bersifat *Close Source* dan

telah menjadi bagian terintegrasi dari *Java Standard Edition* sejak rilis versi JDK 1.1. Kelas-kelas JDBC termuat dalam paket Java.sql. Berawal dari versi 3.0, JDBC kini telah dikembangkan secara pesat dalam *Java Community Process*. JSR 54 mendefinisikan JDBC 3.0 (temuat dalam J2SE(standard *edition*) 1.4). JSR 114 mendefinisikan penambahan JDBC Rowset, dan JSR 221 adalah merupakan spesifikasi dari JDBC 4.0 (termuat dalamJava SE6). JDBC memudahkan berbagai implementasi terhadap bermacam-macam aplikasi yang telah tersedia dan memudahkan pula penggunaan oleh aplikasi yang sama. Oleh API kemudian disediakan mekanisme yang secara dinamis mampu memuat paket Java yang tepat dan mengasosiasikan diri ke JDBCDriver Manager. *Driver Manager* disini berfungsi sebagai sumber koneksi untuk menangani dan membuat seluruh koneksi JDBC. Koneksi JDBC mendukung proses pembuatan dan eksekusi statement. Statement-statement ini dapat berupa statement yang dapat di-*update* seperti INSERT, UPDATE,SQL CREATE, dan DELETE atau berupa statement yang membutuhkan *query* seperti SELECT.

Jenis-jenis statement antara lain:

a. Statement: statement ini dikirim ke *server database* satu persatu dan kontinu setiap saat.
b. *Prepared Statement*: statement ini tersimpan dalam cache yang kemudian jalur eksekusinya telah digolongkan di server database untuk kemudian mampu dieksekusi berulang kali.
c. *Callable Statement*: statement ini digunakan untuk mengeksekusistored procedure di database. Statement-statement *update* seperti INSERT, UPDATE, dan DELETE memberikan nilai *fadback* berupa informasi berapa jumlah baris di database yang telah diperbaharui.

Statement-statement ini tidak memberikan informasi hal yang lain. Lain halnya dengan statement-statement *query*, ia memberikan *fedback* berupa serangkaian hasil baris JDBC. Hasil baris ini digunakan untuk mengetahui nilai-nilai yang terdapat dalam rangkaian hasil. Sedangkan nilai dari tiap-tiap kolom dalam sebuah baris diperoleh dari pendefinisian nama kolom ataupun

nomor kolom yang bersangkutan. Hasil baris juga memiliki metadata yang menjelaskan nama dari masing masing kolom yang mereka bawa dan tipe mereka.

4. *Java Security* Dalam upaya mendukung pembuatan aplikasi yang memiliki tingkat keamanan tinggi, Java menyediakan suatu model pengamanan yang awalnya dikenal sebagai model *sandbox*, model ini pada prinsipnya bertugas untuk membatasi aplikasi *apllet*. Seiring perkembangannya, Java memperbaiki model *sandbox* menjadi fitur-fitur pendukung *security* secara khusus diimplementasikan melalui *API Java Security* dan dicerminkan oleh paket java.security. Paket ini menyediakan koleksi kelas dan *interface* yang mudah untuk dikonfigurasi.
   a. *Provider* Kelas ini mewakili *provider* API Java Security, *provider* menerapkan beberapa atau semua bagiankeamanan Java. Layanan – layanan yang diberikan oleh provider meliputi algoritma kriptografi, pembentukan *key*, konversi dan fasilitas pengelolaan
   b. *Message Digest* Sebagai kriptografi checksum atau *secure hash*. *Message digest* digunakan untuk meningkatkan keamanan transformasi data, seperti *password*. Dalam implemetasinya, nilai message digest diperbandingkan dengan nilai asli. Paket java.security mengimplementasikan *message digest* melalui kelas *MessageDigest*. Untuk menghasilkan *message digest*, menggunakan algoritma MD5 (*Message-Digest algortihm 5*) ialah fungsi hash kriptografik yang digunakan secara luas dengan *hash value128-bit* atau SHA-1(*secure hash algorithm*) adalah fungsi hash kriptografi dirancang oleh *National Security Agency* Amerika Serikat dan diterbitkan oleh NIST Amerika Serikat sebagai *US Federal Information Processing Standard*. SHA - 1 menghasilkan 160 -bit (20 - byte) nilai hash . Sebuah nilai SHA - 1 hash biasanya dinyatakan sebagai angka heksadesimal , 40 angka.
5. *Java Swing*. *Swing* merupakan sebuah teknologi Java untuk pengembangan aplikasi desktop yang dapat berjalan diberbagai macam platform seperti windows, linux, Mac OS X dan Solaris.

6. Java RMI. RMI (*Remote Method Invocation*) adalah cara programmer Java untuk membuat program aplikasi *Java to Java* yang terdistribusi. Program-program yang menggunakan RMI bisa menjalankan metode secara jarak jauh, sehingga program dari server bisa menjalankan method di komputer klien, dan begitu juga sebaliknya. Java RMI yang ada pada bahasa Java telah didesain khusus sehingga hanya bisa bekerja pada lingkungan Java. Hal ini berbeda dengan sistem RMI lainnya, misalnya CORBA, yang biasanya didesain untuk bekerja pada lingkungan yang terdiri dari banyak bahasa dan heterogen. Pemodelan objek pada CORBA tidak boleh mengacu pada bahasa tertentu. Sistem RMI terdiri atas tiga layer/lapisan, yaitu:
   a. Stub/skeleton layer, yaitu stub pada sisi klien (berupa *proxy*), dan skeleton pada sisi server.
   b. *Remote reference layer*, yaitu perilaku *remote reference* (misalnya pemanggilan kepada suatu objek)
   c. *Transport layer*, yaitu *set up* koneksi, pengurusannya dan remote object tracking. Batas antar masing-masing layer disusun oleh interface dan protokol tertentu, yaitu tiap layer bersifat independen terhadap layer lainnya, dan bisa diganti oleh implementasi alternatif tanpa mengganggu layer lainnya. Sebagai contoh, implementasi transport yang digunakan RMI adalah yang berbasis TCP (menggunakan *Java socket*), tapi bisa digantikan dengan menggunakan UDP.
7. *Java 2D/3D*
8. *Java Server Pages* (JSP)
9. *Java Native Interface* (JNI)
10. *Java Sound/Media*
11. *Java Interface Definition Language* (JIDL) +
12. *Common Object Request Broker* (CORBA)
13. *Java Car*
14. *Java Telephony API* (JTAPI)

2.4.2. Eclipse

Eclipse adalah sebuah IDE *(Integrated Development Environment)* untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform *(platform-independent)*.

Berikut ini adalah sifat dari Eclipse:

1. *Multi-platform*: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
2. *Mulit-language*: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lain seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
3. *Multi-role*: Selain sebagai IDE untuk pengembangan aplikasi. Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak seperti dokumentasi, pengujian perangkat lunak, pengembangan web, dan lain sebagainya.

Pada saat ini, Eclipse merupakan salah satu IDE favorit karena gratis dan *open source*. *Open source* berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan membuat komponen yang disebut *plug- in*.

2.4.2.1. Sejarah Eclipse

Eclipse awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak pengembangan IBM Visual Age for Java 4.0. Produk Eclipse ini diluncurkan oleh IBM pada tanggal 5 November 2001. IBM menginvestasikan US$ 40 juta untuk pengembangannya. Sejak 5 November 2001, konsorsium Eclipse Foundation mengambil alih pengembangan Eclipse lebih lanjut.

2.4.2.2. Arsitektur Eclipse

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah *kernel*. Apa

yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah dipasang (diinstal). Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform* (RCP).

Berikut ini adalah komponen yang membentuk RCP:

- *Core platform*
- OSGi
- SWT (*Standard Widget Toolkit*)
- JFace
- *Eclipse Workbench*

Secara standar Eclipse selalu dilengkapi dengan JDT (*Java Development Tools*), *plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in Development Environment*) untuk mengembangkan *plug-in* baru. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE adalah :

1. Terbuka (*open*),
2. Mudah diperluas (*extensible*) untuk apa saja, dan
3. Tidak untuk sesuatu yang spesifik.

2.4.2.3. Versi-versi Eclipse

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran Eclipse secara rutin dan simultan yang dikenal dengan nama *Simultaneous Release*. Setiap versi peluncuran terdiri dari Eclipse Platform dan juga sejumlah proyek yang terlibat dalam proyek Eclipse.

Tujuan sistem ini adalah untuk menyediakan distribusi Eclipse dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah *deployment* dan *maintenance* untuk sistem enterprise, serta untuk kenyamanan. Peluncuran simultan dijadwalkan pada bulan Juni setiap tahunnya.

Tabel 2.9. Versi-versi Eclipse

Kode Peluncuran	Tanggal Peluncuran	Platform	Nama Proyek
Eclipse 3.0	28 Juni 2004	3.0	
Eclipse 3.1	28 Juni 2005	3.1	
Callisto	30 Juni 2006	3.2	Callisto projects
Lanjutan Tabel 2.9. Versi-versi Eclipse			
Europa Europa projects	29 Juni 2007	3.3	
Ganymede	25 Juni 2008	3.4	Ganymede projects
Galileo Galileo projects	24 Juni 2009	3.5	

2.4.3. *MySQL*

2.4.3.1. Pengertian *MySQL*

Kadir (2008:2), "*MySQL* (baca:mai-se-kyu-el) merupakan *software* yang tergolong sebagai *DBMS* (*Database Management System*) yang bersifat *Open Source*."

Sukamto dan Shalahuddin (2013:46), "SQL *(Structured Query Language)* adalah bahasa yang digunakan untuk mengelola data pada RDBMS. SQL awalnya dikembangkan berdasarkan teori aljabar relasional dan kalkulus."

Winarno (2014:102), "*MySQL* adalah sebuah software database. *MySQL* merupakan tipe data relasional yang artinya *MySQL* menyimpan datanya dalam bentuk tabel-tabel yang saling berhubungan."

2.4.3.2. Keunggulan MySQL

Sebagai *software DBMS*, *MySQL* memiliki sejumlah fitur seperti yang dijelaskan di bawah ini:

a. *Multiplatform*

*MySQL*tersedia pada beberapa *platform* (*Windows, Linux, Unix*, dan lain-lain).

b. Andal,cepat, dan mudah digunakan

MySQL tergolong sebagai *database server* (*server* yang melayani permintaan terhadap *database*) yang andal, dpaat menangani *database* yang besar dengan kecepatan tinggi, mendukung banyak sekali fungsi untuk mengaksses*database*, dan sekaligus mudah untuk digunakan.

c. Jaminan keamanan akses

MySQL mendukung pengamanan *database* dengan berbagai kriteria penaksesan. Sebagai gambaran, dimungkinkan untuk mengatur *user* tertentu agar bisa mengakses data yang bersifat rahasia (misalnya gaji pegawai), sedangkan *user* lain tidak boleh.

d. Dukungan *SQL*

Seperti tersirat dalam namanya, *MySQL* mendukung perintah *SQL* (*Structured Query Language*). Sebagaimana diketahui, *SQL* merupakan standar pengaksesan *database* relasional. Pengetahuan akan *SQL* akan memudahkan siapa pun untuk menggunakan *MySQL*.