BAB II

LANDASAN TEORI

2.1. Bencana Alam

Bencana alam sering terjadi di Indonesia. Untuk itu diperlukan pengetahuan serta pemahaman terhadap bencana-bencana alam yang mungkin akan tejadi di masa mendatang. Bencana alam juga menjadi pusat perhatian yang besar dalam menarik dan mengundang respon dari berbagai pihak terhadap para korban bencana. Belajar dari sejumlah bencana yang terjadi di Indonesia, sudah semestinya masyarakat dibekali dengan pengetahuan dan pemahaman tentang bencana, agar mampu menghadapinya ketika diterpa bencana dan turut berperan dalam upaya penanggulangan bencana itu sendiri. Karena bagaimanapun juga disadari bahwa penanggulangan bencana tidak hanya melibatkan pemerintah dan pihak-pihak lain, namun peran masyarakat di dalamnya sangat penting. Setiap orang memiliki definisi sendiri-sendiri dari kata bencana alam. Ada Beberapa pendapat para ahli yang mengungkapkan pengertian tentang bencana alam.

Definisi lain menurut International Strategy for Disaster Reduction (UN-ISDR-2000:24) bencana adalah suatu kejadian yang disebabkan oleh alam atau karena ulah manusia, terjadi secara tiba-tiba atau perlahan-lahan, sehingga menyebabkan hilangnya jiwa manusia, harta benda dan kerusakan lingkungan, kejadian ini terjadi diluar kemampuan masyarakat dengan segala sumber dayanya.

2.1.1. Jenis-Jenis Bencana Alam

Bencana alam, jika ditinjau dari penyebabnya dapat dibagi menjadi tiga jenis yaitu: bencana alam geologis, klimatologis, dan ekstra-terestrial (Buletin

KAMADHIS UGM. 2007:3). Bencana alam geologis adalah bencana alam yang disebabkan oleh gaya-gaya dari dalam bumi. Sedangkan bencana alam klimatologis adalah bencana alam yang disebabkan oleh perubahan iklim, suhu atau cuaca. Bencana alam ekstra-terestrial yaitu bencana alam yang disebabkan oleh gaya atau energi dari luar bumi, bencana alam geologis dan klimatologis yang sering berdampak terhadap manusia.

Jenis Penyebab Bencana Alam	Contoh Kejadian
Bencana alam geologis	Gempa bumi, tsuami, letusan gunung berapi, longsor/gerakan tanah
Bencana alam klimatologis	Banjir, banjir banding, badai, angin putting beliung, kekeringan, kebakaran hutan (bukan oleh manusia)
Bencana alam ekstra-terestrial	Impact/hantaman meteor atau benda dari angkasa luar.

Gambar 2.1 Jenis Bencana Alam (Sumber Buletin KAMADHIS UGM (2007:3))

Bencana alam geologis, terutama gempa bumi, sampai sekarang masih sulit untuk diprediksi, sehingga fenomena alam itu sifatnya mendadak. Namun demikian, peristiwa alam pada dasarnya mempunyai karekteristik umum, yakni gejala awal, gejala utama, dan gejala akhir. Tetapi masalahnya, pada kejadian-kejadian bencana alam geologis, gejala awal tersebut sering kali berjalan terlalu cepat dan berjangka waktu sangat singkat ke gejala utama sehingga tidak ada waktu untuk mengantisipasi datangnya gejala utama. Maka, usaha untuk mendeteksi datanganya gejala awal sangat penting dalam mengantisipasi bencana alam.

Jenis Bencana Alam	Daerah Rawan	Gejala awal
Banjir	Dataran banjir, sempadan, sungai bermeander, lekukan-lekukan di dataran aluvial	Curah hujan tinggi, hujan berlangsung lama, naiknya muka air sungai di stasiun pengamatan
Banjir bandang	Darah bantaran sungai pada transisi datran ke pegunungan	Daerah pegunungan gundul, batuan mudah longsor, curah hujan tinggi, hujan berlangsung lama, terjadi pembendungan di hulu sungai.
Longsor/gerakan tanah	Daerah dengan batuan lepas, batu lempung, tanah tebal, lereng curam.	Curah hujan tinggi, hujan berlangsung lama, munculnya retak-retak pada tanah lereng atas, tiang listrik, pohon, benteng menjadi miring.
Letusan gunung berapi	Lereng dan kaki gunung berapi, terutama yang menghadap kea rah kawah sumbing	Naiknya suhu air kawah, perubahan komposisi kimiawi air dan gas di kawasan guguran Kubah lava, adanya lindu/lini, peningkatan tremor pada seismograf
tsunami	Pantai-pantai yang berhadapan dengan palung tektonik atau gunung api laut	Terjadinya gempa bumi, air laut surut
Gempa bumi	Jalur-jalur tektonik, sesar (patahan) aktif	Peningkatan tremor pada seismograf (yang umumnya sangat singkat kegejala utama)

Gambar 2.2 Gejala Awal Pada Daerah Rawan Bencana (Sumber Buletin

KAMADHIS UGM (2007:3))

2.2. Sistem Informasi

Kata sistem berasal dari bahasa Yunani yaitu systema, yang mempunyai satu pengertian yaitu sehimpunan bagian atau komponen yang saling berhubungan secara teratur dan merupakan satau kesatuan yang tidak terpisahkan (Vaza,2006). Sementara itu menurut Hamalik (2002 dalam Zakir 2007) Sistem secara teknis berarti seperangkat komponen yang saling berhubungan dan bekerja sama untuk mencapai suatu tujuan. Mudyharjo (1993, dalam Zakir 2007) mendefinisikan sistem sebagai suatu kesatuan dari berbagai elemen atas bagian - bagian yang mempunyai hubungan fungsional dan berinteraksi secara dinamis untuk mencapai hasil yang diharapkan.

Dari ketiga definisi tersebut, dapat ditarik kesimpulan bahwa pengertian sistem adalah seperangkat bagian-bagian yang saling berhubungan erat satu dengan lainya untuk mencapai tujuan bersamasama.

Subsistem sebenarnya hanyalah sistem di dalam suatu sistem, sebagai contoh, pesawat terbang adalah suatu sistem yang terdiri dari sistem-sistem bawahan seperti mesin, sistem badan pesawat dan sistem rangka. Masing-masing sistem ini terdiri dari sistem tingkat yang lebih rendah lagi, misal sistem mesin adalah kombinasi dari sistem karburator, sistem bahan bakar dan seterusnya. Istilah subsistem digunakan untuk memudahkan analisis dan pengkomunikasian.

Berikut ini adalah karakter atau sifat-sifat tertentu yang dimiliki oleh sistem :

1. Mempunyai Komponen (*Component*).

Suatu sistem mempunyai sejumlah komponen yang saling berinteraksi dan bekerjasama untuk membentuk suatu kesatuan.Setiap komponen mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan.

2. Batas Sistem (*Boudary*).

Batas sistem merupakan daerah yang membatasi antara suatu sitem dengan sistem lainya.

3. Penghubung Sistem (*Interface*).

Penghubung merupakan media antara subsistem dengan subsistem lainya. Penghubung memungkinkan sumber-sumber daya mengalit dari satu subsistem ke subsistem lainya, dan juga subsistem -subsistem tersebut dapat berintegrasi membentuk satu kesatuan.

4. Masukan Sistem (*Input*).

Sesuatu yang dimasukan ke dalam sistem yang berasal dari lingkungan.

5. Keluaran Sistem (*Output*).

Suatu hasil dari proses pengolahan sistem yang dikeluarkan ke lingkungan

6. Pengolahan Sistem (*proces*).

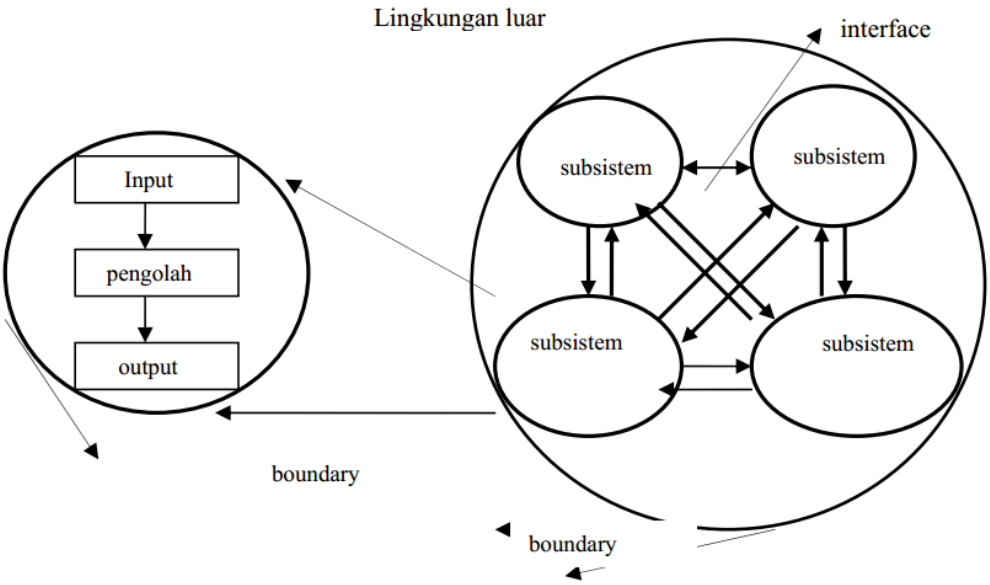
Suatu sistem dapat mempunyai suatu bagian pengolahan yang akan mengubah masukan menjadi keluaran.

7. Linkungan Luar Sistem (*Enviroments*).

Segala sesuatu di luar batas suatu sistem yang mempengaruhi kerja sistem.

8. Sasaran Suatu Tujuan (*Goal*).

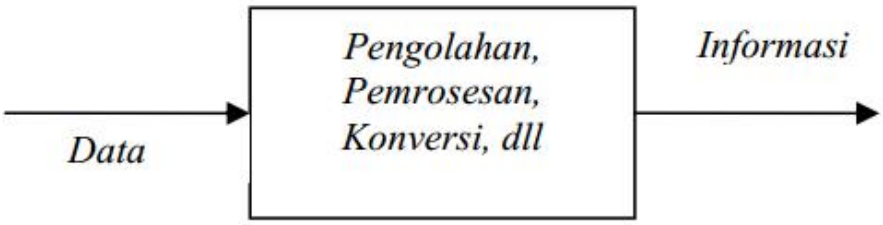
Setiap sistem mempunyai tujun. Suatu sistem dikatakan berhasil jika mengenai sasaran atau tujuan (goal).



Gambar 2.4 Karakteristik Sistem

2.3. Data dan Informasi

Seringkali istilah informasi dan data agak rancu karena kedua istilah tersebut sering digunakan secara bergantian dan saling tertukar, meskipun kedua istilah ini sebenarnya merujuk pada masing - masing konsep yang berbeda. Data merupakan bahasa mathematical dan simbol-simbol pengganti lain yang disepakati oleh umum dalam menggambarkan objek, manusia, peristiwa, aktivitas, konsep dan objek-objek penting lainya., data merupakan suatu kenyataan apa adanya (raw facts). Sedangkan informasi adalah data yang ditempatkan pada konteks yang penuh arti oleh penerimanya (John, 1983 dalam Prahasta, 2002).



Gambar 2.5 Hubungan Data dan Informasi

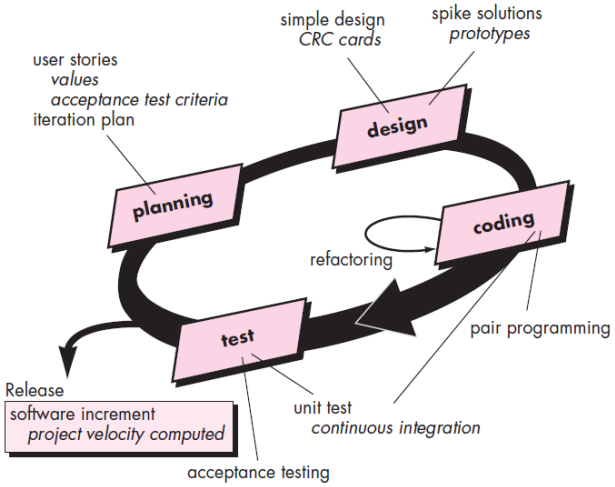
2.4. Metode Pengembangan Sistem *Extreme Programming (XP)*

2.4.1. Metode

Metode adalah suatu cara atau teknik yang sistematis untuk mengerjakan sesuatu. Metode penelitian adalah prosedur-prosedur, konsep-konsep pekerjaan, aturan-aturan yang akan digunakan oleh suatu ilmu pengetahuan.

2.4.2. Metode Pengembangan Sistem

Extreme programming adalah metode pengembangan perangkat lunak yang ringan dan termasuk salah satu *agile methods* yang dipelopori oleh Kent Beck, Ron Jeffries, dan Ward Cunningham. *Extreme programming* merupakan salah satu metodologi dalam rekayasa perangkat lunak dan juga merupakan satu dari beberapa *agile software development methodologies* yang berfokus pada *coding* sebagai aktivitas utama disemua tahap pada siklus pengembangan perangkat lunak (*software development lifecycle*). Metodologi ini mengedepankan proses pengembangan yang lebih responsive terhadap kebutuhan *costumer (agile)* dibandingkan dengan metode-metode tradisional dalam membangun suatu *software* dengan kualitas yang lebih baik.



Gambar 2.6 Model Proses *Extreme Programming*
(Pressman, 2010)

Model proses *extreme programming* terbentuk dari sebuah kerangka kerja yang memiliki 4 konteks aktivitas utama yaitu *planning*, *design*, *coding* dan *testing*. Berikut ini adalah beberapa aktivitas dalam metode *extreme programming*:

1. *Planning*

Aktivitas planning pada model proses XP berfokus pada mendapatkan gambaran fitur serta fungsi dari perangkat lunak yang akan dibangun dengan mengumpulkan semua bahan data-data dan kebutuhan dari pengguna.

2. *Design*

Aktivitas design dalam pengembangan aplikasi bertujuan untuk mengatur pola logika dalam sistem. Sebuah design yang baik dapat mengurangi ketergantungan antar setiap proses pada sebuah sistem. Dengan begitu, jika salah satu fitur pada sistem mengalami kerusakan, tidak akan mempengaruhi sistem secara keseluruhan. Dalam XP, proses *design* terjadi sebelum dan sesudah

aktivitas *coding* berlangsung. Yang berarti aktivitas *design* terjadi secara terus-menerus selama proses pengembangan aplikasi berlangsung.

3. *Coding*

Pada tahap ini pengembang aplikasi memilih *prototype* dan kemudian dilakukan pembuatan sistem dengan menggunakan suatu bahasa pemrograman tertentu atau mengembangkan sistem yang sudah ada menjadi sistem baru.

4. *Test*

Tahapan uji coba pada XP sudah dilakukan juga pada saat tahapan sebelumnya, yaitu *coding*. XP menerapkan perbaikan masalah kecil dengan sesegera mungkin akan lebih baik dibandingkan menyelesaikan masalah pada saat akan mencapai tenggat akhir. Oleh karena itu setiap modul yang sedang dikembangkan akan terlebih dahulu mengalami pengujian dengan modul unit tes yang telah dibuat sebelumnya.

Setelah semua modul telah dikumpulkan dalam sebuah sistem yang sempurna, barulah pengujian penerimaan (*acceptance test*) dilakukan. Pada tahapan ini aplikasi langsung diuji coba oleh pengguna dan mendapat tanggapan langsung mengenai penerapan skema yang telah digambarkan sebelumnya.

Terdapat keunggulan pada metode XP, diantaranya:

1. Mampu menyelesaikan suatu proyek perangkat lunak dengan cepat.
2. Setiap orang dalam tim bisa mengerjakan tiap tahapan dalam metode ini, tanpa harus menunggu tahapan lain selesai.
3. Kemungkinan resiko kegagalan dalam suatu proyek menjadi lebih kecil.
4. Setiap kebutuhan pelanggan dapat diselesaikan dengan cepat dan baik.

5. Dalam pengimplementasian, pelanggan dapat ikut serta dalam hal pengujian kelayakan dan kebutuhan aplikasi yang dibangun.

2.5. Pemrograman Berorientasi Objek (PBO)

Menurut Rickyanto (2004), PBO merupakan paradigma pemrograman yang populer saat ini yang telah menggantikan teknik pemrograman berbasis prosedur. *Object oriented programming* (OOP) yang berarti pula pemrograman berorientasi objek sudah ditemukan sekitar tahun 1960 dan dikembangkan pada permulaan tahun 1970. Pemrograman berorientasi objek *(object oriented programming /OOP*) merupakan pemrograman yang berorientasikan kepada objek, dimana semua data dan fungsi dibungkus dalam *class-class* atau objek-objek. Setiap objek dapat menerima pesan, memproses data, mengirim, menyimpan dan memanipulasi data. Beberapa objek berinteraksi dengan saling memberikan informasi satu terhadap yang lainnya.

Masing-masing objek harus berisikan informasi mengenai dirinya sendiri dan dapat dihubungkan dengan objek lain. Pemrograman berorientasi objek berbeda dengan pemrograman prosedural yang hanya menggunakan satu halaman ke bawah untuk mengerjakan banyak perintah atau statement. Penggunaan pemrograman berorientasi objek sangat banyak sekali, contoh: C#, VB.Net, java, php, perl dan lainnya. Dalam konsep pemrograman berorientasi objek dikenal beberapa istilah umum, yaitu:

1. *Attribute*

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat memiliki hak akses *private, public* maupun *protected*. Sebuah

atribut yang dinyatakan sebagai *private* hanya dapat diakses secara langsung oleh kelas yang membungkusnya, sedangkan kelas lainnya tidak dapat mengakses atribut ini secara langsung. Sebuah atribut yang dinyatakan sebagai *public* dapat diakses secara langsung oleh kelas lain di luar kelas yang membungkusnya. Sebuah atribut yang dinyatakan sebagai *protected* tidak dapat diakses secara langsung oleh kelas lain di kelas yang membungkusnya, kecuali kelas yang mengaksesnya adalah kelas turunan dari kelas yang membungkusnya. Karakteristik atribut dalam *class* disebut sebagai *variable*. Pada atribut, terdapat pula dua istilah *variable*, yaitu *instance variable* dan *class variable*.

Pada *instance variable,* tiap objek memiliki *instance variable* dan menyimpan nilainya tersendiri. Sedangkan pada *class variable,* atribut yang dimiliki oleh semua objek yang berasal dari *class* yang sama, serta semua objek memiliki nilai *class variable* yang sama, atribut yang dimiliki objek dari class yang sama.

2. *Method*

Method adalah fungsi atau prosedur yang dibuat oleh seorang *programmer* dalam suatu *class*. Dengan kata lain, *method* pada sebuah kelas (*class)* hampir sama dengan fungsi atau prosedur pada pemrograman prosedural. Pada sebuah *method* di dalam sebuah kelas juga memiliki izin akses seperti halnya atribut pada kelas, izin akses itu antara lain *private, public* dan *protected* yang memilki arti sama pada izin akses atribut yang telah dibahas sebelumnya.

Sebuah kelas boleh memiliki lebih dari satu *method* dengan nama yang sama asalkan memilki parameter masukan yang berbeda sehingga *compiler* atau

interpreter dapat mengenali *method* mana yang dipanggil. Hal ini dinamakan *overloading*. *Method* merupakan serangkaian *statement*/perintah (perintah = baris program) dalam suatu *class* yang meng-*handle task* tertentu. *Method* merupakan hal-hal yang bisa dilakukan oleh objek lain dari suatu *class method* didefinisikan pada *class* akan tetapi dipanggil melalui objek.

3. *Class*

Merupakan model yang berisi kumpulan *attribute* dan *method* dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh, *class* manusia memiliki *attribute* berat, tinggi, usia kemudian memiliki *method* makan, minum, tidur. *Method* dalam sebuah *class* dapat merubah *attribute* yang dimiliki oleh *class* tersebut. Sebuah *class* merupakan dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. *Class* didefinisikan sebagai sebuah *blue-print* (denah) atau *prototype* yang mendefinisikan varibel-variabel dan metode-metode yang umum untuk semua objek dari *n* jenis tertentu (*n* maksudnya jumlah tertentu). Sebuah kelas menyerupai sebuah struktur yang merupakan tipe data sendiri, misalkan tipe data titik yang terdiri dari koordinat *x* dan *y*.

Class adalah *template* untuk pembuatan objek. Karakterisitik *class* memilki beberapa karakterisitik, diantaranya: anggota *class* terdiri dari *attribute* dan *method*. Tiap-tiap anggota *class* memilki control pengaksesan tersendiri, maksundnya adalah apakah anggota *class* tersebut dapat diakses dengan bebas (dengan tipe *public*) atau hanya dapat diakses melalui sebuah *interface*. Dalam hal ini, *interface* adalah *device* yang digunakan untuk komunikasi antar objek berbeda

yang tidak memiliki hubungan apapun. *Interface* bisa dikatan sebagai *protocol* komunikasi anatar objek tersebut.

4. Objek

Objek merupakan perwujudan dari *class*, setiap objek akan mempunyai *attribute* dan *method* yang dimiliki oleh *class*-nya, contohnya: Amir, Ahmad, Yani merupakan objek dari *class* manusia. Setiap objek dapat berinteraksi dengan objek lainnya meskipun berasal dari *class* yang berbeda. Objek merupakan sesuatu yang memilki identitas (nama), pada umumnya juga memiliki data tentang dirinya maupun objek lain dan mempunyai kemampuan untuk melakukan sesuatu dan bisa bekerja sama dengan objek lainnya. Objek adalah implementasi dari *class* yang secara sederhanaya, dapat dikatakan terdiri dari *property* (atribut) dan *method*. Karakteristik objek memiliki dua karakteristik utama, yaitu *attribute* dan *behavior*. *Attribute* merupakan status objek dan *behavior* merupakan tingkah laku dari objek tersebut.

Dalam pengembangan perangkat lunak berorientasi objek, objek dalam perangkat lunak akan menyimpan *state*-nya dalam variable dan menyimpan informasi tingah laku (*behavior*) dalam *method-method* atau fungsi-fungsi/prosedur.

Karakteristik atau sifat-sifat yang dipunyai oleh pemrograman berorientasi objek adalah sebagai berikut:

1. Abstraksi

Abstraksi adalah prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

2. Enkapsulasi

Enkapsulasi adalah pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

3. Pewarisan (*inheritance*)

Pewarisan merupakan mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh defines dan objek lain sebagai bagian dari dirinya.

4. *Reusability*

Reusability merupakan pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

5. Generalisasi dan Spesialisasi

Menunjukan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.

6. Komunikasi Antar Objek

Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirim dari satu objek ke objek lainnya.

7. *Polymorphism*

Polymorphism adalah kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

2.6. Metode Pengumpulan Data

Metode pengumpulan data dilakukan untuk memperoleh informasi yang dibutuhkan dalam rangka mencapai tujuan penelitian dalam pembangunan sistem informasi titik rawan bencana ini. Teknik pengumpulan data yang dilakukan adalah sebagai berikut: (Pressman, 2002)

1. Wawancara (interview)

   Metode dilakukan dengan wawancara atau konsultasi secara langsung dengan narasumber dengan menyiapkan pertanyaan-pertanyaan yang sesuai dengan kebutuhan untuk memperoleh data-data yang dibutuhkan.

2. Pengamatan (Obervasi)

   Metode dilakukan dengan pengamatan langsung pada objek penelitian di lapangan atau suatu kegiatan untuk mempelajari objek yang dipilih dan untuk memperoleh data-data yang relevan dan akurat akan digunakan untuk perancangan aplikasi tersebut.

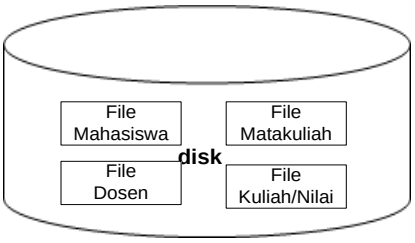
3. Studi Kepustakaan

   Metode dilakukan dengan cara mengumpulkan data, bahan-bahan tertulis dengan cara membaca buku-buku referensi dan media lain yang berhubungan dengan pembahasan masalah-masalah yang dihadapi dalam perancangan aplikasi tersebut.

2.7. Basis Data (*Database*)

2.7.1. Pengertian Basis Data

Basis data (*database*) merupakan kumpulan data yang saling berhubungan satu dengan yang lainnya tersimpan di simpanan luar komputer dan digunakan perangkat lunak untuk memanipulasinya. *Database* merupakan salah satu komponen yang penting dalam pertukaran data, karena berfungsi sebagai basis penyedia data bagi para pemakainya.



Gambar 2.7 Basis Pada Media Penyimpanan Perangkat Keras
(Ringgodoni, 2011)

Adapun istilah-istilah dalam basis data diantaranya adalah sebagai berikut:

1) *Entity* merupakan individu yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain, misalnya orang, tempat, kejadian, atau konsep yang informasinya direkam.
2) *Attribute* merupakan sesuatu atau hal yang mendeskripsikan karaktersistik (*property*) dari *entity*.
3) *Data value* (nilai atau isi data) adalah data actual atau informasi yang disimpan pada *attribute*.
4) *Record* (tupel) yaitu kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu entity secara lengkap.

5) File merupakan kumpulan record-record sejenis yang mempunyai panjang elemen yang sama, *attribute* yang sama, namun berbeda-beda *data value*-nya.
6) *Database management system* (DBMS) merupakan kumpulan *file* yang saling berkaitan bersama dengan program untuk mengolahnya. *Database* merupakan kumpulan datanya, sedangkan program pengelolaannya berdiri sendiri dalam satu paket program yang berfungsi untuk membaca data, mengisi data, menghapus data serta melaporkan data dalam *database*.

2.7.2. Kriteria-Kriteria Basis Data

Basis data (*database*) memiliki beberapa kriteria yang harus dipenuhi, yaitu sebagai berikut:

1) Bersifat *data oriented* dan bukan *program oriented*.
2) Dapat digunakan oleh beberapa program aplikasi tanpa perlu mengubah basis datanya.
3) Dapat berkembang dengan mudah, baik volume maupun strukturnya.
4) Dapat memenuhi sistem-sistem baru secara mudah.
5) Dapat digunakan dengan cara-cara berbeda.
6) Kerangkapan (redudansi) data dapat diminimalkan.

2.8. Alat Pemodelan Sistem

2.8.1. *Unified Modeling Language* (UML)

Unified modeling language adalah keluarga notasi grafis yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (PBO). UML merupakan standar yang relative terbuka yang dikontrol oleh *Object Management Group* (OMG), sebuah konsorium terbuka yang terdiri dari banyak perusahaan. OMG dibentuk untuk membuat standar-standar yang mendukung interoperabilitas, khususnya interoperabilitas sistem berorientasi objek. Beberapa jenis diagram dalam UML yang sering digunakan ditunjukan pada tabel berikut ini:

Tabel 2.1 Beberapa Jenis Diagram UML

Diagram	Kegunaan
Use case	Bagaimana pengguna berinteraksi dengan sebuah sistem.
Class	*Class*, fitur dan hubungan-hubungan.
Sequence	Interaksi antar objek, penekanan pada *sequence*.
Activity	*Behavior procedural* dan *parallel*.
Deployment	Pemindah artifak ke node.

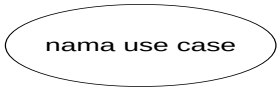
1. *Use Case Diagram*

Use case adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use case* mendeskripsikan interaksi tipikal antara para pengguna sistem

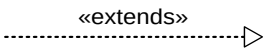
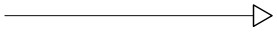
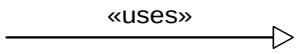
dengan sistem itu sendiri, dengan memberikan sebuah narasi tentang bagaimana sistem tersebut digunakan.

Dalam bahasa *use case*, para pengguna disebut sebagai actor. *Actor* merupakan sebuah peran yang dimainkan seorang pengguna dalam kaitannya dengan sistem. *Actor* dapat meliputi pelanggan, petugas layanan konsumen, manajer penjualan dan analis produk. Seorang *actor* dapat menggunakan banyak *use case*, sebaliknya, sebuah *use case* juga dapat digunakan oleh beberapa *actor*. *Actor* tidak harus manusia. Jika sebuah sistem melakukan sebuah layanan untuk sistem komputer lain, sistem lain tersebut merupakan *actor*. Berikut ini merupakan simbol-simbol yang digunakan dalam *use case diagram*:

Tabel 2.2 Simbol-Simbol *Use Case Diagram*

Simbol	**Deskripsi**
Use case 	Fungsioanalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau actor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*.
Aktor/*actor* 	Orang, proses atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri; biasanya dinyatakan menggunakan kata benda di awal frase nama actor.

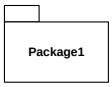
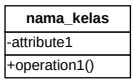
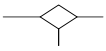
Tabel 2.2 Simbol-Simbol *Use Case Diagram* (Lanjutan)

Asosiasi/ *association*  	Komunikasi antara actor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan actor.
Ekstensi/*extend*  	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buat *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Include/uses  	Include berarti *use case* yang ditambahkan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan.

2. *Class Diagram*

Class diagram mendeskripsikan jensi-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat di antara objek-objek tersebut. *Class* diagram juga menunjukan property dan operasi sebuah *class* dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut. UML menggunakan istilah fitur sebagai istilah umum yang meliputi property dan operasi sebuah *class*. Berikut ini simbol-simbol yang digunakan pada *class* diagram:

Tabel 2.3 Simbol-Simbol *Class Diagram*

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas.
Kelas 	Kelas pada struktur sistem
Antarmuka/*interface*	Sama dengan konsep interface dalam pemrograman berorientasi objek.
Asosiasi /*association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity.
Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / *dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas.
Agregasi/*aggregation* 	Relasi antar kelas dengan makna semua bagian (*whole-part*)

3. *Sequence Diagram*

Interaction diagram menunjukkan bagaimana kelompok-kelompok objek saling berkolaborasi dalam beberapa behavior. UML memiliki beberapa bentuk interaction diagram dan yang paling umum digunakan adalah *sequence diagram*. Sebuah sequence diagram, secara khusus, menjabarkan behavior sebuah scenario tunggal. Diagram tersebut menunjukan sejmlah objek contoh dan pesan-pesan

yang melewati objek-objek ini di dalam use case. Berikut ini merupakan simbol-simbol dalm sequence diagram:

Tabel 2.4 Simbol-Simbol *Sequence Diagram*

Simbol	**Deskripsi**
Actor  Atau 	Orang, proses atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluat sistem yang akan dibuat itu sendiri; biasanya dinyatakan menggunakan kata benda di awal frase nama actor.
Garis hidup / *life line* 	Arah panah mengarah pada objek yang memilki 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 ojek yang menerima kembalian.

4. *Activity Diagram*

Activity diagram adalah teknik untuk menggambarkan logika procedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, diagram ini memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara diagram ini dan notasi diagram alir adalah diagram ini mendukung *behavior paralel*.

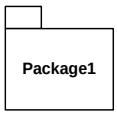
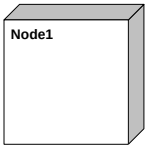
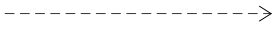
Tabel 2.6 Simbol-Simbol *Activity Diagram*

Simbol	**Deskripsi**
Status awal 	Status awal aktivitas 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 stau aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

5. *Deployment Diagram*

Deployment diagram menunjukkan susunan fisik sebuah sistem, menunjukkan bagian perangkat lunak mana yang berjalan pada perangkat keras mana. Selain itu, diagram ini menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi.

Tabel 2.7 Simbol-Simbol *Deployment Diagram*

Simbol	**Deskripsi**
Package 	Package merupakan sebuah bungkusan dari satu atau lebih node.
Node 	Biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika dalam node disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen.
Kebergantungan/ *dependency* 	Kebergantungan antar node, arah panah mengarah pada node yang dipakai.
Link 	Relasi antar node.

2.9. *Black Box Testing*

Pengujian adalah suatu set aktifitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan. Pada pengujian aplikasi yang akan dibuat nantinya menggunakan *black box testing* (pengujian kotak hitam). *Black-box tesing* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukkan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan (Rosa dan Salahudin, 2011).

Black box testing berusaha untuk menemukan kesalahan dalam ketegori sebagai berikut:

1) Fungsi-fungsi yang salah atau hilang.
2) Kesalahan interface.
3) Kesalahan dalam struktur data atau akses database eksternal.
4) Kesalahan performa.
5) Kesalahan inisialisasi dan terminasi.

2.10. Google Maps

Google Maps adalah layanan gratis yang diberikan oleh Google dan sangat popular. Google Maps adalah suatu peta dunia yang dapat kita gunakan untuk melihat suatu daerah. Dengan kata lain, Google Maps merupakan suatu peta yang dapat dilihat dengan menggunakan suatu browser. Kita dapat menambahkan fitur Google Maps dalam web yang telah kita buat atau pada blog kita yang berbayar maupun gratis sekalipun dengan Google Maps API. Google Maps API adalah suatu library yang berbentuk JavaScript.

2.10.1. Jenis Peta Pilihan Google Maps

Pada Google Maps terdapat 4 jenis pilihan model peta yang disediakan oleh Google, diantaranya adalah:

1. *ROADMAP*, ini yang saya pilih, untuk menampilkan peta biasa 2 dimensi
2. *SATELLITE*, untuk menampilkan foto satelit
3. *TERRAIN*, untuk menunjukkan relief fisik permukaan bumi dan menunjukkan seberapa tingginya suatu lokasi, contohnya akan menunjukkan gunung dan sungai
4. *HYBRID*, akan menunjukkan foto satelit yang diatasnya tergambar pula apa yang tampil pada ROADMAP (jalan dan nama kota).

2.11. *Hypertext Preprocessing* (PHP)

PHP adalah suatu bahasa pemrograman *open source* yang digunakan secara luas terutama untuk pengembangan web dan dapat disimpan dalam bentuk html. Untuk menghasilkan sebuah HTML, script yang ditulis menggunakan PHP mempunyai perintah yang lebih singkat dibandingkan bahasa pemrograman lain seperti Perl atau C. Hanya perlu memasukankode untuk melakukan sesuatu diantara tag awal dan tag akhir PHP.

Keuntungan utama menggunakan PHP adalah *script* PHP tidak hanya benar – benar sederhana bagi pemula, tetapi juga menyediakan banyak fitur tambahan untuk *programmer* profesional. Jangan ketakutan membaca daftar fitur PHP yang panjang. Anda dapat melewatinya dan mulai menulis *script* sederhana dalam beberapa jam.

Script PHP dapat digunakan dalam tiga hal yaitu :

1. Penulisan program *Server Side*. Hal ini adalah target utama PHP. Diperlukan tiga hal agar *script* PHP dapat bekerja antara lain, PHP Parser (CGI atau *server module*), *server web* (misal apache) dan browser web.
2. Penulisan program *Command Line*. Script PHP dapat berjalan tanpa server atau browser. Hanya diperlukan PHP parser dalam bentuk *Command Line*.
3. Penulisan program untuk aplikasi desktop.

Sebelum mulai membuat program PHP, diperlukan perangkat sebagai berikut :

1. Editor teks atau web *development tool* seperti Notepad, PHPed, vi, emac, Ultra edit, Dreamweaver, dan lainnya.
2. Untuk melihat dan melakukan pengujian pada suatu halaman, diperlukan browser web (IE 5.0 atau lebih, Netscape Navigator, dan lainnya).
3. Server yang mendukung PHP.

Secara singkat, kelebihan PHP meliputi :

1. Script PHP sederhana, mudah dibuat, dan mempunyai kecepatan akses tinggi.
2. Dapat berjalan dalam server web yang berbeda dan dalam sistem operasi yang berbeda. PHP dapat berjalan pada sistem operasi Linux/Unix, Windows, dan Macintosh.
3. Bersifat *Open Source* sehingga diterbitkan secara gratis.
4. Dapat berjalan pada server web Microsoft Personal Web Server, Apache, IIS, Xitamidan sebagainya.

5. Termasuk bahasa yang *embedded* (bisa ditempel atau diletakkan dalam tag HTML).

2.12. *Web Server*

Web Server adalah sebuah perangkat lunak server yang berfungsi menerima permintaan HTTP atau HTTPS dari klien yang dikenal dengan web browser dan mengirimkan kembali hasilnya dalam halaman-halaman web yang umumnya berbentuk dokumen HTML". Web server yang dimaksud disini adalah simulasi dari sebuah web server secara fisik. Web server biasanya juga disebut HTTP server karena menggunakan protocol HTTP sebagai basisnya. Beberapa web server yang sering digunakan diantaranya adalah PWS, IIS, Apache dan sebagainya.

2.13. Metode Pengujian *Black Box*

Menurut Perry (2006), Black-box *testing* adalah metode pengujian perangkat lunak yang melakukan pengetesan fungsionalitas dari aplikasi yang bertentangan dengan struktur internal atau kerja (pengujian white-box). Pengetahuan khusus dari kode aplikasi atau struktur internal dan pengetahuan pemrograman pada umumnya tidak diperlukan. Uji kasus dibangun di sekitar spesifikasi dan persyaratan, yakni, apa yang seharusnya aplikasi lakukan. Menggunakan deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus. Tes ini dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Perancang uji memilih input yang valid dan tidak valid dan menentukan *output* yang benar.

Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan masukan. Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit *testing* juga.

Metode ujicoba blackbox memfokuskan pada keperluan fungsional dari *software*. Karena itu ujicoba blackbox memungkinkan pengembang *software* untuk membuat himpunan kondisi *input* yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba blackbox bukan merupakan alternatif dari ujicoba whitebox, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode whitebox.

Ujicoba blackbox berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses *database* eksternal
4. Kesalahan performa
5. Kesalahan inisialisasi dan terminasi