BAB II

LANDASAN TEORI

2.1. Tinjauan Teori Umum

2.1.1. Sistem

Menurut Sutabri, (2012: 6) sistem adalah sekelompok unsur yang erat hubungannya satu dengan yang lain, yang berfungsi bersama-sama untuk mencapai tujuan tertentu.

Gordon B. Davis dalam Sutabri (2012: 6) menyatakan bahwa sistem bisa berupa abstrak atau fisik. Sistem yang abstrak adalah susunan gagasan-gagasan atau konsepsi yang teratur yang saling bergantung. Sedangkan sistem yang bersifat fisik adalah serangkaian unsur yang bekerja sama untuk mencapai suatu tujuan.

2.1.1.1. Karakteristik Sistem

1. Komponen Sistem (*Components*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk suatu kesatuan.

2. Batasan Sistem (*Boundary*)

   Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem lainnya atau sistem dengan lingkungan luarnya. Batasan

sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkungan Luar Sistem (*Environment*)

   Bentuk apapun yang ada diluar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut dengan lingkungan luar sistem.

4. Penghubung sistem (*Interface*)

   Media yang menghubungkan sistem dengan subsistem yang lain disebut dengan penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem yang lain. Keluaran suatu subsistem akan menjadi masukan untuk subsistem yang lain dengan melewati penghubung.

5. Masukan Sistem (*Input*)

   Energi yang dimasukan kedalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal (*signal input*).

6. Keluaran Sistem (*Output*)

   Hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain.

7. Pengolah Sistem (*Procces*)

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

8. Sasaran Sistem (Objective)

   Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministik. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan.

2.1.2. Informasi

Menurut Sutabri, (2012: 21), informasi merupakan proses lebih lanjut dari data yang sudah memiliki nilai tambah. Nilai dari informasi ditentukan dari 2 (dua) hal, yaitu manfaat dan biaya untuk mendapatkannya. Sementara itu menurut Sutanta (2011: 13), informasi merupakan hasil pengolahan data sehingga menjadi bentuk yang penting bagi penerimanya dan mempunyai kegunaan sebagai dasar dalam pengambilan keputusan yang dapat dirasakan akibatnya secara langsung saat itu juga untuk atau secara tidak langsung pada saat mendatang. Untuk memperoleh informasi, diperlukan data yang akan diolah dan unit pengolah

2.1.2.1. Nilai dan Kualitas Informasi

Nilai dari informasi ditentukan dari 2 (dua) hal, yaitu manfaat dan biaya untuk mendapatkannya. Suatu informasi dikatakan bernilai apabila manfaat yang diperoleh lebih berharga dibandingkan dengan biaya untuk mendapatkannya. Keuntungan dari sebagian informasi tidak dapat dihitung dengan suatu nilai uang tetapi dapat ditaksir nilai efektifitasnya. Nilai informasi biasanya dihubungkan

dengan analisis *cost effectiveness* atau *cost benefit*. Nilai informasi ini didasarkan atas 10 sifat, yaitu :

1. Mudah diperoleh

   Sifat ini menunjukkan kemudahan dan kecepatan untuk memperoleh informasi.

2. Luas dan lengkap

   Sifat ini menunjukkan kelengkapan isi informasi. Hal ini tidak hanya mengenai volumenya, akan tetapi juga mengenai keluaran informasinya. Sifat ini sangat kabur dan karena itu sulit untuk mengukurnya.

3. Ketelitian

   Sifat ini berhubungan dengan tingkat kebebasan dari kesalahan keluaran informasi. Pada volume data yang besar biasanya terdapat dua jenis kesalahan, yakni kesalahan pencatatan dan kesalahan perhitungan.

4. Kecocokan

   Sifat ini menunjukkan seberapa baik keluaran informasi dalam hubungannya dengan permintaan para pemakai. Isi informasi harus ada hubungannya dengan masalah yang sedang dihadapi sedangkan semua keluaran yang lainnya tidak berguna. Sifat ini sulit untuk diukur.

5. Ketepatan waktu

   Sifat ini berhubungan dengan waktu yang dilalui, yang lebih pendek dari siklus untuk mendapatkan informasi. Masukkan, pengolahan, dan pelaporan keluaran kepada para pemakai, biasanya tepat waktu. Dalam beberapa hal, ketepatan waktu dapat diukur.

6. Kejelasan

   Sifat ini menunjukkan tingkat kejelasan informasi. Informasi hendaknya terbebas dari istilah-istilah yang tidak jelas.

7. Keluwesan

   Sifat ini berhubungan dengan apakah informasi tersebut dapat digunakan untuk membuat lebih dari satu keputusan, tetapi juga apakah dapat digunakan untuk lebih dari seorang pengambil keputusan. Sifat ini sulit diukur akan tetapi dalam beberapa hal dapat diukur dengan suatu nilai tertentu.

8. Dapat dibuktikan

   Sifat ini menunjukkan sejauh mana informasi itu dapat diuji oleh beberapa pemakai hingga sampai didapatkan kesimpulan yang sama.

9. Tidak ada prasangka

   Sifat ini berhubungan dengan ada tidaknya keinginan untuk mengubah informasi tersebut guna mendapatkan kesimpulan yang telah diarahkan sebelumnya.

10. Dapat diukur

    Sifat ini menunjukkan hakikat informasi yang dihasilkan oleh sistem informasi formal.

Kualitas dari suatu informasi tergantung dari 3 (tiga) hal, yaitu informasi harus akurat (*accurate*), tepat waktu (*timelines)*, dan relevan (*relevance*).

1. Akurat (*accurate*)

   Informasi harus bebas dari kesalahan dan tidak biasa atau menyesatkan. Akurat juga berarti bahwa informasi harus jelas mencerminkan maksudnya. Informasi harus akurat karena dari sumber informasi sampai ke penerima informasi mungkin banyak mengalami gangguan (*noise*) yang dapat mengubah atau merusak informasi tersebut.

2. Tepat waktu (*timelines*)

   Informasi yang sampai kepada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi, karena informasi merupakan landasan didalam pengambilan keputusan. Bila pengambilan keputusan terlambat maka dapat berakibat fatal bagi organisasi.

3. Relevan (*relevance*)

   Informasi tersebut mempunyai manfaat untuk pemakainya. Relevansi informasi untuk setiap orang berbeda.

2.1.3. Sistem Informasi

Menurut Sutabri (2012: 46), sistem informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan.

Dalam arti yang luas, sistem informasi dapat dipahami sebagai sekumpulan subsistem yang saling berhubungan, berkumpul bersama-sama dan membentuk

satu kesatuan, saling berinteraksi dan bekerja sama antara bagian satu dengan yang lainnya dengan cara-cara tertentu untuk melakukan fungsi pengolahan data, menerima masukan (*input*) berupa data-data, kemudian mengolahnya (*processing*), dan menghasilkan keluaran (*output*) berupa informasi sebagai dasar bagi pengambilan keputusan yang berguna dan mempunyai nilai nyata yang dapat dirasakan akibatnya baik pada saat itu juga maupun di masa mendatang, mendukung kegiatan operasional, manjeril, dan strategis organisasi, dengan memanfaatkan berbagai sumber yang ada dan tersedia bagi fungsi tersebut guna mencapai tujuan (Sutanta 2011: 16).

Berdasarkan komponen fisik penyusunnya, sistem informasi terdiri atas komponen berikut:

1. Perangkat keras (*hardware)*

   Perangkat keras dalam sistem informasi meliputi perangkat-perangkat yang digunakan oleh sistem komputer untuk masukan dan keluaran *(input/output device), memory, modem,* pengolah *(processor),* dan periferal lainnya.

2. Perangkat lunak (*software*)

   Perangkat lunak dalam sistem informasi adalah berupa program-program komputer yang meliputi sistem operasi (*Operating System/OS*), bahasa pemrograman (*programming language*), dan program-program aplikasi (*application*).

3. Berkas basis data (*file*)

   Berkas merupakan sekumpulan data dalam basis data yang disimpan dengan cara-cara tertentu sehingga dapat digunakan kembali dengan mudah dan cepat.

4. Prosedur (*procedure*)

   Prosedur meliputi prosedur pengoperasian untuk sistem informasi, manual, dan dokumen-dokumen yang memuat aturan-aturan yang berhubungan dengan sistem informasi dan lainnya.

5. Manusia (*brainware*)

   Manusia yang terlibat dalam suatu sistem informasi meliputi *operator, programmer, system analyst,* manajer sistem informasi, manajer pada tingkat operasional, manajer pada tingkat manajerial, manajer pada tingkat strategis, teknisi, administrator basis data (*Database Administrator*/DBA), serta individu lain yang terlibat di dalamnya.

Spesialis informasi (*information specialist*) menggambarkan pegawai perusahaan yang bertanggung jawab penuh untuk mengembangkan dan memelihara sistem informasi berbasis komputer (*Computer Based Information Systems/CBIS*). Spesialis informasi digolongkan menjadi lima macam, yaitu:

1. Analis sistem

   Analis sistem adalah seseorang pakar yang mampu mendefenisikan masalah dan menyiapkan dokumentasi tertulis mengenai cara komputer membantu pemecahan masalah. Analis sistem bekerja sama dengan pemakai mengembangkan sistem baru dan memperbaiki sistem yang ada sekarang.

2. Pengelola basis data (*Database Administrator/DBA*)

   DBA bekerja sama dengan pemakai dan analis sistem basis data yang berisi data yang diperlukan untuk menghasilkan informasi bagi pemakai.

3. Spesialis jaringan (*network specialist*)

   Spesialis jaringan adalah orang yang ahli dalam bidang komputer dan telekomunikasi. Spesialis jaringan bekerja sama dengan analis sistem dan pemakai membentuk jaringan komunikasi data yang menyatukan berbagai sumber daya komputer yang tersebar.

4. Pemrogram (*programmer*)

   Pemrogram bekerja dengan menggunakan dokumentasi yang disiapkan oleh analis sistem untuk membuat kode program dalam bahasa tertentu untuk memproses data masukan yang tersedia menjadi keluaran berupa informasi bagi para pemakai.

5. Operator

   Operator mengoperasikan peralatan komputer berskala besar (misalnya *mainframe*, mini), memantau layar komputer, mengganti ukuran kertas di *printer*, mengelola pustaka *disk storage*, dan lain-lain.

2.1.4. SDLC (*Software Development Life Cycle*)

SDLC atau *Software Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan untuk

mengembangkan sistem-sistem perangkat lunak sebelumnya. Tahapan-tahapan yang ada pada SDLC secara global menurut Rosa (2011: 24-26) adalah sebagai berikut:

1. Inisiasi (*initiation*)

   Tahap ini biasanya ditandai dengan pembuatan proposal proyek perangkat lunak. Pengembangan konsep sistem (*system concept development*) Mendefinisikan lingkup konsep termasuk dokumen lingkup sistem, analisis manfaat biaya, manajemen rencana, dan pembelajaran kemudahan sistem.

2. Perencanaan (*planning*)

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

3. Analisa kebutuhan (*requirments analysis*)

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

4. Desain (*design*)

   Mentransformasikan kebutuhan *detail* menjadi kebutuhan yang sudah lengkap, dokumen *design* sistem fokus pada bagaimana dapat memenuhi fungsi-fungsi yang dibutuhkan.

5. Pengembangan (*development*)

   Mengkonversi *design* ke sistem informasi yang lengkap termasuk bagaimana memperoleh dan melakukan instalasi lingkungan sistem yang dibutuhkan. Membuat basis data dan mempersiapkan prosedur kasus pengujia, mempersiapkan berkas atau *file* pengujian, pengodean, pengompilasian, memperbaiki dan membersihkan program, peninjauan pengujian.

6. Integrasi dan pengujian (*intergration and test*)

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

7. Implementasi (*implementation*)

   Termasuk pada persiapan implementasi, implementasi perangkat lunak pada lingkungan produksi (lingkungan pada *user*) dan menjalankan resolusi dari permasalahan yang terindentifikasi dari fase integrasi dan pengujian.

8. Operation dan pemeliharaan (*Operations and maintenance*)

   Mendeskripsikan pekerjaan untuk mengoperasikan dan memelihara sistem informasi pada lingkungan produksi, termasuk implementasi akhir dan masuk pada proses peninjauan.

9. Disposisi (*disposition*)

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

Hal terpenting adalah bagaimana mengenali tipe pelanggan (*customer*) dan memilih menggunakan model SDLC yang sesuai dengan karakter pelanggan dan sesuai dengan karakter pengembang.

SDLC memiliki beberapa model dalam penerapan tahapan prosesnya, berikut adalah model-model SDLC menurut (Rosa 2011: 26-37):

1. Model *Waterfall*

   Model SDLC air terjun (*waterfall*) sering juga disebut model sekuensal linier (*sequential linier*) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, *design*, pengodean, pengujian, dan tahap pendukung (*support*).

2. Model *Prototype*

   Model *prototype* dimulai dari pengumpulkan kebutuhan pelanggan terhadap perangkat lunak yang akan dibuat. Lalu dibuatlah program *Prototype* agar pelanggan lebih terbayang dengan apa sebenarnya dinginkan. Program *Prototype* biasanya merupakan program *Prototype* yang belum jadi. Program ini biasanya merupakan program yang belum jadi.

3. Model *Rapid Application Development* (RAD)

   Rapid Application Development (RAD) adalah model proses pengembangan perangkat lunak yang bersifat inkremental terutama untuk waktu pengerjaan yang pendek. Model RAD adalah adaptasi dari model air terjun untuk mengembangkan setiap komponen perangkat lunak.

4. Model *Iteratif*

   Model iteratif mengkombinasikan proses-proses pada model air terjun dan iteratif pada *prototype*. Model inkremental akan menghasilkan versi-versi perangkat lunak yang sudah mengalami penambahan fungsi untuk setiap pertambahannya (inkremen/*increment*).

5. Model *Spiral*

   Model spiral memasangkan iteratif dan model *prototype* dengan *control* dan aspek sistematik yang diambil dari model air terjun.

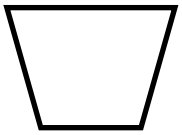
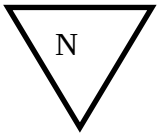
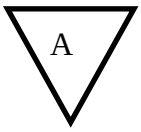
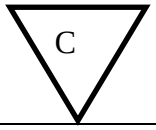
Dari kelima model SDLC yang sudah dijelaskan diatas maka penulis akan menggunakan model *waterfall* karena pengaplikasian menggunakan model ini mudah dan tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan dari analisis, desain, *coding*, dan *testing*.

2.1.5. Bagan alir (*Flowchart*)

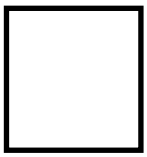
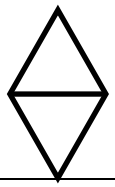
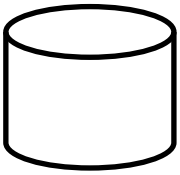
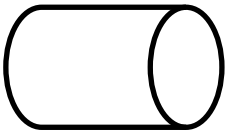
Bagan alir (*flowhcart*) adalah bagian bagan (*chart*) yang menunjukkan alir (*flow*) di dalam program atau prosedur sistem secara logika. Bagan alir digunakan

terutaman untuk alat bantu komunikasi dan untuk dokumentasi (Jogiyanto,2005: 795).Bagan alir sistem digambarkan dengan menggunakan simbol-simbol sebagai berikut ini:

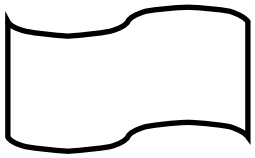
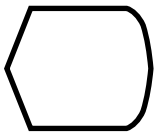
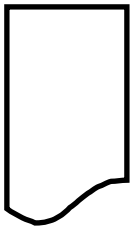
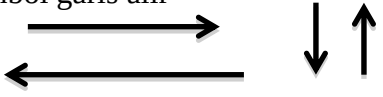
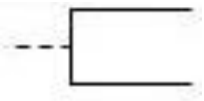
Tabel 2.1. Bagan alir *sistem* (*flowchart*)

Simbol	**Keterangan**
Simbol dokumen 	Menunjukkan dokumen *input* dan *output* baik untuk proses manual, mekanik atau computer
Simbol kegiatan manual 	Menunjukkan pekerjaan manual
Simbol simpanan *offline* 	*File non* komputer yang di arsip urut angka (*numerical*)
	*File non* komputer yang di arsip urut huruf (*alfabetical*)
	*File non* komputer yang di arsip urut tanggal (*cronological*)
Simbol kartu plong 	Menunjukkan *input/output* yang menggunakan kartu plong (*purchase card*)
Simbol proses 	Menunjukkan kegiatan proses dari operasi program computer

Tabel 2.1. Lanjutan

Simbol operasi luar 	Menunjukkan operasi yang dilakukan diluar proses computer
Simbol pengurutan *offline* 	Menunjukkan proses pengurutan data diluar proses computer
Simbol pita *magnetic* 	Menunjukkan *input/output*menggunakan pita *magnetik*
Simbol hard disk 	Menunjukkan *input/output*menggunakan*hard disk*
Simbol disket 	Menunjukkan*input/output* menggunakan *diskette*
Simbol drum magnetic 	Menunjukkan *input/ouput* menggunakan *drum magnetik*

Tabel 2.1. Lanjutan

Simbol pita kertas berlubang 	Menunjukkan *input/output* menggunakan pita kertas berlubang
Simbol *keyboard* 	Menggunakan*input* yang menggunakan *online keyboard*
Simbol *diplay* 	Menunjukkan *output* yang ditampilkan monitor
Simbol pita control 	Menunjukkan penggunaan pita *control* dalam *batch control total* untuk pencocokan diproses *batch processing*
Simbol hubungan komunikasi 	Menunjukkanproses transmisi data melalui *channel* komunikasi
Simbol garis alir 	Menunjukkan arus dari proses
Simbol penjelasan 	Menunjukkan penjelasan dari suatu proses

Tabel 2.1. Lanjutan

Simbol penghubung 	Menunjukkan penghubung kehalaman yang masih sama atau kehalaman lain

Sumber: Jogiyanto (2005: 797)

2.1.6. UML (Unifed Modeling Languange)

UML (*Unifed Modeling Langunge*) adalah salah satu alat bantu yang sangat handal didunia pengembangan sistem berorientasi obyek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan *visual* yang memungkinkan bagi pengembang membuat cetak biru atas visi mereka dalam bentuk yang baku (Munawar, 2005: 17). Sedangkan menurut (Rosa, 2011: 120) UML adalah sekumpulan spesifikasi yang dikeluarkan oleh *Object Management Group* (OMG) yang terdiri dari UML *Superstructure*, dan *Object Constraint Language* (OCL).

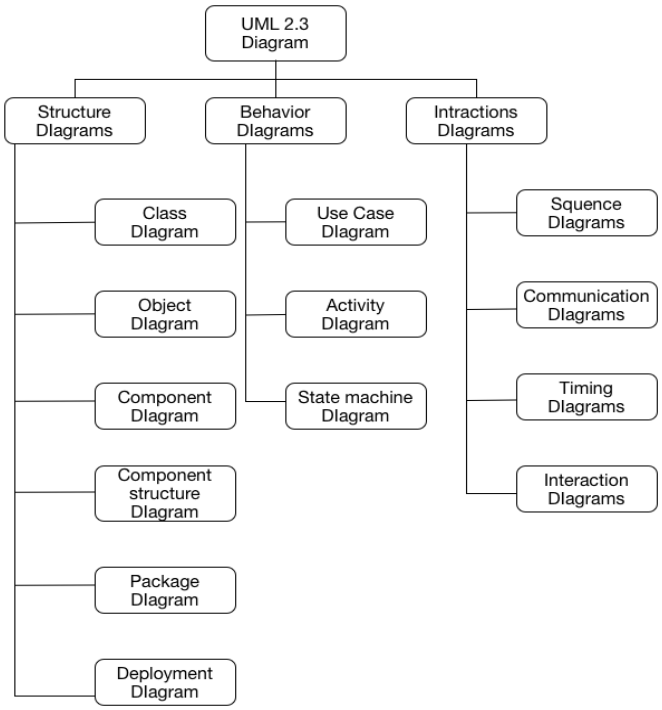
Menurut Yasin (2012: 194) *Unified Modelling Language* (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Gambaran dari UML yaitu sebagai berikut:

1. UML sebagai Bahasa Pemodelan UML merupakan bahasa pemodelan yang memiliki pembendaharaan kata dan cara untuk mempresentasikan secara fokus pada konseptual dan fisik dari suatu sistem.

2. UML sebagai bahasa untuk menggambarkan sistem (*Visualizing*) UML tidak hanya merupakan rangkaian simbol grafikal, cukup dengan tiap simbol pada notasi UML merupakan penetapan simantik yang baik. UML merupakan suatu model eksplisit yang menggambarkan komunikasi informasi pada sistem.
3. UML sebagai bahasa untuk menspesifikasikan sistem (*Specifying*) Maksudnya membangun model yang sesuai, tidak ambigu dan lengkap. Pada faktanya, UML menunjukkan semua spesifikasi keputusan analisis, desain dan implementasi yang penting yang harus dibuat pada saat pengembangan dan penyebaran dari sistem software intensif.
4. UML sebagai bahasa untuk membangun sistem (*Constructing*) UML bukan bahasa pemrograman visual, tetapi model UML dapat dikoneksikan secara langsung pada bahasa pemrograman visual.
5. UML sebagai bahasa untuk pendokumentasian sistem (*Documenting*) Maksudnya UML menunjukkan dokumentasi dari arsitektur sistem dan *detail* dari semuanya. UML hanya memberikan bahasa untuk memperlihatkan permintaan dan untuk tes. UML menyediakan bahasa untuk memodelkan aktifitas dari perencanaan *project* dan manajemen pelepasan (*release management*).

2.1.6.1. Diagram UML

Diagram UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.1 dibawah ini:



Gambar 2.1. Diagram UML

Berikut ini penjelasan singkat dari pembagian kategori tersebut:

1. *Structure* diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior* diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakukan sistem atau rangkain perubahan yang terjadi pada sebuah sistem.

3. *Interaction diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antara subsistem pada suatu sistem.

Dari 13 diagram diatas disini peniliti hanya menggunakan *Class diagram*, *Use Case diagram*, *Activity diagram* dan *Squance diagram*, berikut penjelasan dari masing-masing diagram:

1. *Class Diagram*

   Diagran kelas atau *class diagram* menggambarkan struktur sistem dari pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas Memiliki atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas, sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan *cohesion* dan *coupling*. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam suatu sebuah kelas.

Berikut adalah simbol-simbol yang ada pada diagram kelas:

Tabel 2.2. *Class diagram* simbol

Simbol	Deskripsi
Kelas **nama_kelas** +atribut +operasi()	Kelas pada struktur system
Antarmuka / *Interface* **nama_interface**	Sama dengan konsep *interface* dalam pemograman berorientasi objek
Asosiasi / *association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *ditected association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi(umum khusus)
Kebergantungan / *dependency*	Relasi antara kelas dengan makna kebergantungan antar kelas
Agregasi / *aggregation*	Relasi antar kelas dengan makna semua bagian (*whol –part*)

Sumber: Rosa (2011 : 123)

2. *Use Case Diagram*

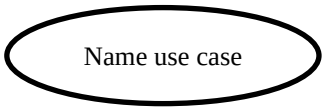
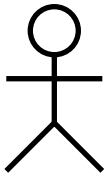
Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

Aktor: merupakan orang, proses, atau sistem lain 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.

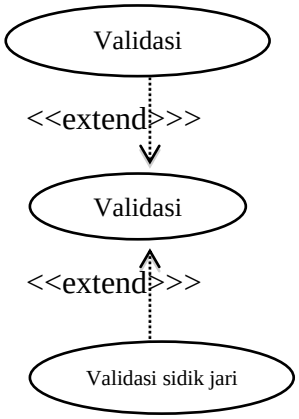
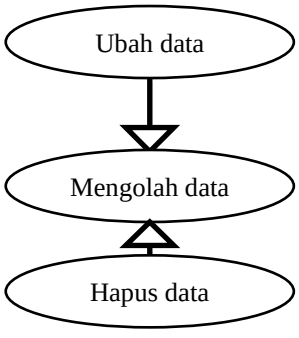
Use case: merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut adalah simbol-simbol yang ada pada diagram *use case*:

Tabel 2.3. *Use case* simbol

Simbol	Deskripsi
Use case 	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antara unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal-awal frase nama *use case.*
Aktor / *actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun *symbol* dari aktor adalah gambar orang, biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
Asosiasi / *association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.

Tabel 2.3. Lanjutan

Ektensi / *extend* <<extend>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan yaitu, mirip dengan prinsip *inheritance*pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan sama dengan *use case* yang ditambahkan missal Arah panah mengarah pada *use case* yang ditambahkan 
Generalisasi / *generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainya, misalnya: 

Tabel 2.3. Lanjutan

	Arah panah mengarah pada *use case* yang menjadi generalisasi (umum)
Menggunakan / include / *uses* <<include>>  <<uses>> 	Relasi *use case* tambah ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini Ada dua sudut pandang yang cukup besar mengenai *include* di *usecase*: Include berarti use case yang ditambahkan akan selalu dipanggil saat *use case* yang ditambahkan dijalankan Include berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan

Sumber: Rosa (2011: 135)

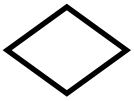
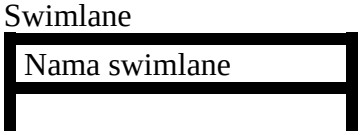
3. Activity Diagram

Diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.

2. Urutan atau pengelompokan tampilan dari *sistem/userinterface* dimana setiap aktivitas di anggap memiliki sebuah rencanngan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan khusus ujinya.

Tabel 2.4.*Activity* diagram simbol

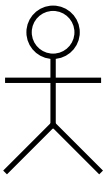
Simbol	**Deskripsi**
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan / *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber: Rosa (2011 : 135)

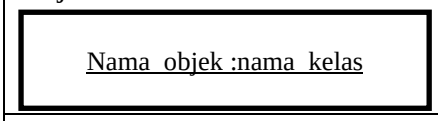
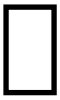
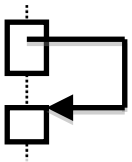
4. Sequence Diagram

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

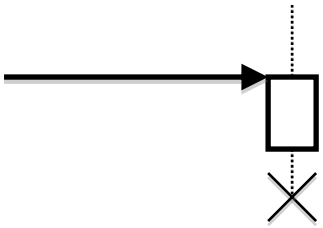
Tabel 2.5.*Sequence* diagram simbol

Simbol	**Deskripsi**
Aktor  Nama_aktor Atau Nama_aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama actor
Garis hidup / *lifeline* 	Menyatakan kehidupan suatu objek

Tabel 2.5. Lanjutan

Objek  	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan
Pesan tipe create «create» 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tipe *call* 1:nama_metode 	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* 1: masukan 	Menyatakan bahwa suatu objek mengirim data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi

Tabel 2.5. Lanjutan

Pesan tipe *return* 1:keluaran 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe *destroy* «destory» 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*

Sumber: Rosa (2011: 138-139)

2.2. Tinjauan Teori Khusus

2.2.1. Indekos

Dalam kamus besar bahasa Indonesia indekos berarti tinggal di rumah orang lain dengan atau tanpa makan dengan membayar setiap bulan (kbbi.web.id 19/10/2016).

Rumah kost merupakan salah satu tempat penyedia jasa penginapan atau tempat tinggal sementara yang terdiri dari beberapa kamar dan setiap kamar memiliki beberapa fasilitas yang ditawarkan atau disediakan dan juga mempunyai harga yang telah ditentukan oleh pemilik kost, sedangkan lama waktu penyewaan ditentukan sendiri oleh si penyewa kamar (Andry Rachmadi, 2013).

2.2.2. Android

Menurut Firdan Ardiansyah (2011: 4) Android adalah sistem operasi untuk telepon seluler yang berbasiskan Linux. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri sehingga dapat digunakan oleh bermacam peranti penggerak. Awalnya Google Inc. membeli Android Inc. pendatang baru yang membuat *software* (perangkat lunak) untuk telepon genggam.

Pencetus gagasan lahirnya Android dimulai oleh Google Inc. yang berkolaborasi dengan Android Inc. Android Inc. adalaah perusahaan yang berada di Palo Alto, California Amerika Serikat, merupakan pendatang baru yang

membuat peranti lunak untuk ponsel. Kemudian untuk mengambangkan dan mempercanggih sistem operasi Android, maka dibentuklah *Open Handset Alliance*, konsorium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi. Perusahaan tersebut diantaranya Google, INTEL, HTC, Motorola, Qualcomm, T-Mobile.

Menurut Lee (2012: 2), Android merupakan sistem operasi ponsel yang didasari oleh Linux. Android awalnya dibuat dari suatu proyek yang bernama sama yaitu Android.Inc. Pada tahun 2005, sebagai strategi untuk merambah dunia ponsel, Google membeli proyek Android juga tim yang membuatnya. Android disenangi oleh masyarakat karena fleksibilitasnya, dan disenangi oleh para pembuat aplikasi karena ketika membuat aplikasi untuk Android, maka aplikasi itu dapat dijalankan pada semua ponsel bermerek apapun, selama ponsel itu mempunyai sistem operasi Android.

2.2.2.1. Fitur-fitur Android

Karena Android bersifat *open-source* dan dapat digunakan oleh pembuat aplikasi secara bebas, tidak ada pengaturan resmi untuk perangkat keras dan perangkat lunak untuk Android. Walaupun begitu, Android mempunyai *fitur-fitur* tersendiri seperti yang diutarakan oleh Lee (2012: 3), antara lain:

1. Penyimpanan Data, menggunakan SQLite, sebuah basis data yang ringan untuk penyimpanan data.

2. Konektifitas, mendukung fitur GSM/EDGE, IDEN, CDMA, EVDO, UMTS, Bluetooth (termasuk A2DP dan AVRCP), Wi-Fi, LTE, dan WiMAX.
3. Pengiriman Pesan, mendukung SMS dan MMS.
4. *Web Browser*, berbasis pada aplikasi *open-source* WebKit dengan menggunakan mesin Chrome's V8 JavaScript.
5. Media, mendukung media H.263, H.264 (dalam wujud 3GP atau MP4), MPEG-4 SP, AMR, AMR-WB (dalam wujud 3GP), AAC, HE-AAC (dalam wujud MP4 atau 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, dan BMP. Perangkat Keras, dilengkapi sensor Accelerometer, Kamera, Kompas Digital, Sensor Jarak, dan GPS.
6. *Multi-Touch*, mendukung fitur layar *multi-touch*.
7. *Multi-Tasking*, mendukung aplikasi yang mempunyai fitur *multitask.*
8. Flash, Android versi 2.3 mendukung Flash versi 10.1.
9. Tethering, mendukung fitur berbagi koneksi internet sebagai *hotspot* tanpa kabel.

Fitur akan terus bertambah seiring dengan terus berkembangnya teknologi.

2.2.3. Javascript

Javascript merupakan bahasa pemrograman untuk membuat *website* di internet dan dapat bekerja di sebagian besar *browse*r. Javascript berkerja bersama dengan HTML5 untuk membuat sebuah logika pemrograman serta membuat website menjadi lebih beragam. Javascript memungkinkan objek pada web untuk

berinteraksi dengan pengguna, mengontrol *browserweb*, dan mengubah isi dokumen yang muncul dalam layar *webbrowser*. Javascript merupakan bahasa pemrograman pada web. Mayoritas *website* saat ini menggunakan Javascript. Javascript merupakan bahasa pemrograman tingkat tinggi, dinamis yang sangat cocok untuk pemrograman berbasis objek (*object-oriented*) dan fungsi (Flanagan, 2011).

Penggunaan Javascript pada perancangan ini dikarenakan aplikasi yang dirancang ini berbasis *web* aplikasi.

2.2.4. Angular

Merujuk (AngularJS, 2015) AngularJS yaitu AngularJS merupakan *framework* javascript berbasis *open-source* yang dirilis oleh Google pada tahun 2009 dengan tagline berikut ini***"HTML Enhanced for Web apps!"***yang di Maksud dari *tagline* AngularJS ini adalah HTML yang ditingkatkan fungsinya untuk membangun *web app*. Melihat sejarah kemunculan HTML, awalnya HTML hanya digunakan untuk membuat dokumen statis (website) bukan untuk membuat *web app.* Nah, sekarang bayangkan kalau sejak awal HTML memang dikembangkan untuk membuat *web app*, seperti itulah konsep AngularJS. AngularJS bukan merupakan pustaka (*library*) *javascript* melainkan sebuah *framework* yang solid untuk membangun *web app*, seperti *framework javascript* pada umumnya AngularJS mengadopsi konsep MVC (*Model, View, Controller*), meskipun menggunakan implementasi yang berbeda dengan konsep asli MVC.

Angular digunakan dalam perancangan ini karena merupakan pendukung dari Javascript. Dengan penggunaan Angular proses pembuatan aplikasi menjadi lebih cepat.

2.2.5. PHP

Menurut Saputra (2012: 2) PHP atau yang memiliki kepanjangan PHP *Hypertext Preprocessor*, merupakan suatu bahasa pemrograman yang difungsikan untuk membangun suatu website dinamis. PHP menyatu dengan kode HTML, maksudnya adalah beda kondisi, HTML digunakan sebagai pembangun atau pondasi dari kerangka layout web, sedangkan PHP difungsikan sebagai prosesnya, sehingga dengan adanya PHP tersebut, sebuah web akan sangat mudah di-maintenance.

Menurut (Peranginangin, 2006: 2–3) PHP singkatan dari PHP *Hypertext Preprocessor* yang digunakan sebagai bahasa *script server-side* dalam pengembangan web yang disisipkan pada dokumen HTML. PHP diciptakan pertama kali oleh Rasmus Lerdorf pada tahun 1994. Awalnya, PHP digunakan untuk mencatat jumlah serta untuk mengetahui siapa saja pengunjung pada *homepagenya*. Menurut Sibero (2014: 49) PHP adalah pemrograman interpreter yaitu proses penerjemahan baris kode sumber menjadi kode mesin yang dimengerti komputer secara langsung pada saat baris kode dijalankan. PHP disebut sebagai pemrograman *Server Side Programming*, hal ini dikarenakan seluruh prosesnya dijalankan pada server. PHP adalah suatu bahasa dengan hak

cipta terbuka atau yang juga dikenal dengan istilah *Open Source*, yaitu pengguna dapat mengembangkan kode-kode fungsi PHP sesuai dengan kebutuhannya.

Pemrograman PHP dapat ditulis dalam dua bentuk yaitu penulisan baris kode PHP pada file tunggal dan penulisan kode PHP pada halaman html (*embedded*). Kedua cara penulisan tersebut tidak memiliki perbedaan, hanya menjadi kebiasaan gaya penulisan dari programmer. PHP memiliki banyak kelebihan yang tidak dimiliki oleh bahasa *script* sejenisnya. PHP difokuskan pada pembuatan *script srever-side*, yang bisa melakukan apa saya yang dapat dilakukan oleh CGI, seperti mengumpulkan data dari *form*, menghasilkan isi halaman *web* dinamis, dan kemampuan mengirim serta menerima *cookies*, bahkan lebih daripada kemampuan CGI.

PHP dapat digunakan pada semua sistem informasi, antara lain Linux, Unix (termasuk variannya HP-UX, Solaris, dan OpenBSD), Microsoft Windows, Mac OS X, RISC OS. PHP juga mendukung banyak *web server* seperti Apache, *Microsoft Internet Information Server* (MIIS), Personal *Web Server* (PWS), Netscape and iPlanet servers, Oreilly *Website* Pro *server*, Audium, Xitami, OmniHTTPd, dan masih banyak lagi lainnya, bahkan PHP dapat bekerja sebagai suatu CGI Processor.

Salah satu fitur yang dapat diandalkan oleh PHP adalah dukungannya terhadap banyak *database*. *Database* yang dapat didukung oleh PHP adalah Adabas D, dBase, Direct MS-SQL, Empress, FilePro (read only), FrontBase, Hyperwave, IBM DB2, dan Informix.(Peranginangin, 2006: 2-3)

Penggunaan PHP pada perancangan ini dikarenakan php dapat digunakan oleh semua sistem informasi sehingga mempermudah proses perancangan.

2.2.6. Laravel

McCool (2012: 15) menyatakan bahwa Laravel adalah *framework* MVC (*Model View Controller*) yang dikembangkan dengan bahasa pemograman PHP. Laravel sendiri dirancang untuk meningkatkan kualitas perangkat lunak yang dibangun untuk menyederhanakan tugas-tugas yang digunakan sehingga menjadi lebih praktis seperti *otentikasi, routing, sessions* dan *caching*, sehingga dengan mengurangi biaya pengembangan dan pemeliharaan.

Laravel menjadi pilihan bagi peneliti dalam merancang aplikasi ini, dikarenakan Laravel merupakan sebuah *framework* yang mendukung PHP.

2.2.7. MySQL

Menurut Peranginangin (2006: 381) MySQL adalah suatu *Relational Database Management System* (RDBMS) yang mendukung *database* yang terdiri dari sekumpulan relasi atau *table*. Relasi dan *table* memiliki arti yang sama. Menurut Wahyudi (2008: 243), SQL adalah bahasa non prosedural untuk mengakses *database*. SQL adalah standar bahasa komputer versi ANSI (*American National Standards Institute*) untuk mengakses dan memanipulasi sistem *database*. Perintah-perintah di SQL digunakan untuk mengambil kembali (*retrieve*) dan meperbarui (*update*) data di *database*. SQL dapat bekerja sama

dengan program *database* lain seperti MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase dan sebagainya.

Menurut Sibero (2014: 97) MySQL adalah suatu RDBMS (*Relational Database Management System*) yaitu aplikasi sistem yang menjalankan fungsi pengolahan data. MySQL pertama dikembangkan oleh MySQL AB yang kemudian diakuisisi Sun Microsystem dan terakhir dikelola oleh *Oracle Coorporation.*

2.2.8. Google Maps

Google Maps adalah sebuah jasa peta globe virtual gratis dan online yang disediakan oleh Google, google maps ini dapat diakses di http://maps.google.com/. Google Map API merupakan aplikasi interface yang dapat diakses lewat *javascript* agar Google Map dapat ditampilkan pada halaman web yang sedang kita bangun. Untuk dapat mengakses Google Map, pengguna harus melakukan pendaftaran *Api Key* terlebih dahulu dengan data pendaftaran.

Google Maps API adalah sebuah layanan (*service*) yang diberikan oleh Google kepada para pengguna untuk memanfaatkan Google Map dalam mengembangkan aplikasi. Google Maps API menyediakan beberapa fitur untuk memanipulasi peta, dan menambah konten melalui berbagai jenis *services* yang dimiliki, serta mengijinkan kepada pengguna untuk membangun aplikasi *enterprise* di dalam websitenya.

Google Maps menjadi pilihan bagi peneliti dikarenakan goolge maps dapat diakses lewat *javascript* agar Google Map dapat ditampilkan pada halaman web.