BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Untuk mendukung penelitian ini, maka diperlukan tinjauan pustaka yang diambil dari beberapa jurnal penelitian terdahulu.

Oleh Titin Kristina (2018) jurusan Manajemen Informatika, AMIK BSI Jakarta dengan judul Sistem Pendukung Keputusan Dengan Menggunakan Metode TOPSIS Untuk Pemilihan Lokasi Pendirian Grosir Pulsa. Penelitian ini membahas tentang sistem pendukung keputusan dengan menggunakan metode topsis untuk pemilihan lokasi pendirian grosir pulsa. Penelitian ini bertujuan untuk membangun sebuah sistem penunjang keputusan yang berfungsi sebagai alat bantu bagi para wirausahawan dalam mengambil keputusan memilih lokasi yang tepat pendirian grosir pulsa. Penelitian ini menggunakan metode topsis. Sebelum penerapan metode topsis, penulis melakukan survey untuk memastikan detai kriteria yang akan digunakan. Kriteria yang digunakan berjumlah lima yaitu lokasi strategis, kepadatan penduduk sekitar lokasi, pendapatan masyarakat sekitar lokasi, dekat dengan sarana umum, dan tingkat keamanan yang mendukung. Serta penulis memilih tiga alternatif lokasi yang akan dijadikan grosir pulsa yaitu Karawaci, Kotabumi, dan Serpong. Setelah dilakukan perhitungan secara manual menggunakan metode topsis, alternatif dengan kode A2 yaitu Kotabumi memiliki nilai referensi tertinggi dengan nilai preferensi 0,666 , lebih tinggi dibandingkan dengan alternatif A1 dengan nilai 0,286 dan alternatif A3 dengan nilai 0,420. Sehingga lokasi yang paling sesuai adalah Kotabumi, karena paling sesuai dengan kriteria dari perusahaan yakni lokasi strategis, kepadatan penduduk sekitar lokasi, pendapatan masyarakat sekitar lokasi, dekat dengan sarana umum, dan tingkat keamanan yang mendukung.

Oleh Avriana Indarwasti, Barizana S.A, Prottasof G.K (2017) jurusan Teknik Informatika dan Komputer, Politeknik Negri Jakarta dengan judul Sistem Pendukung Keputusan Pemilihan Tempat Kuliner Di Depok Dengan TOPSIS. Di wilayah Depok terdapat banyak pilihan wisata kuliner serta fashion, akan tetapi tidak banyak para wisatawan kesulitan mencari informasi tempat-tempat yang direkomendasikan.Dengan permasalahn tersebut, maka penulis menyarankan

sebuah penelitian membuat sebuah sistem pendung keputusan pemilihan tempat kuliner di Depok dengan TOPSIS yang menyediakan informasi kuliner berbasis web. Digunakan TOPSIS yang dapat melakukan perangkingan menurut kelebihan tempat kuliner dengan kriteria yang sama. Kriteria penilaian yang digunakan adalah rasa, variasi menu, waktu operasional, area dan fasilitas dimana tiap kriteria akan diberikan bobot. Hasil yang diperoleh dari penelitian ini adalah alternatif saran atau pilihan tempat kuliner terbaik. Dari hasil perhitungan dengan menggunakan TOPSIS didapatkan tempat kuliner WP menempati urutan pertama dengan nilai preferensi relatif untuksetiap alternatifnya sebesar 0,77. OTW menempati urutan kedua, dengan nilai preferensi relatifnya adalah 0,66. WU sebagai urutan ketiga dengan nilai preferensi relatif 0,63.

Oleh Muhamad Aries Permana, Anggi Srimurdianti Sukamto, Tursina (2017) jurusan Teknik Informatika, Universitas Tanjungpura dengan judul Rancang Bangun Aplikasi Sistem Pendukung Keputusan Pemilihan Gedung Lokasi Pernikahan Menggunakan Metode Promethee (Studi Kasus: Pontianak). Penelitian selanjutnya ini membahas tentang sebuah rancang bangun aplikasi sitem pendukung keputusan pemilihan lokasi gedung pernikahan menggunkana metode PROMETHEE dengan studi kasus di Pontianak. Tujuan dari penelitian ini adalah untuk menghasilkan sebuah aplikasi yang dapat membantu pasangan yang akan melangsungkan pernikahan, dalam memilih dan memilah gedung tanpa perlu survei secara langsung menuju lokasi gedung. Metode Promethee menggunakan kriteria dan bobot dari masing-masing kriteria yang kemudian diolah untuk menentukan pemilihan alternatif lapangan, yang hasilnya berurutan berdasarkan prioritasnya. Degan kriteria yang disajikan yaitu biaya sewa, kapasitas tamu, luas bangunan, luas parkir, dan lama sewa. Dari hasil kombinasi perhitungan dengan metoode promethee, maka diperoleh urutan penentuan lokasi gedung pernikahan berdasarkan alternatifnya yaitu, urutan pertama adalah A1, urutan kedua adalah A2, urutan ketiga adalah A3, urutan keempat adalah A4, urutan kelima adalah A5, dan urutan keenam adalah A6.

Oleh Putri Alit Widyastuti Santiary, Putu Indah Ciptayani, Ni G. A. Harry Saptarini, I Ketut Swardika (2018) jurusan Teknik Elektro, Politeknik Negeri Bali dengan judul Sistem Pendukung Keputusan Penentuan Lokasi Wisata Dengan

Metode TOPSIS. Penelitian ini bertujuan untuk membangun sistem pendukung keputusan (SPK) untuk penentuan lokasi wisata dengan metode TOPSIS dan fuzzy. Metode ini akan memberikan pembobotan kriteria sesuai dengan kondisi/preferensi pengguna, dan kemudian melakukan pengolahan pada data yang bersifat rasa/fuzzy. Metode TOPSIS akan memberikan perankingan alternatif yang menjamin kedekatan dengan kriteria benefit dan menjauhkannya dari kriteria yang bersifat cost. Implementasi sistem dilakukan dengan menggunakan database MySQL dan bahasa PHP. SPK yang dibangun mampu menghasilkan rekomendasi dengan memberikan perankingan lokasi wisata kepada pengguna sesuai preferensinya. Sistem yang dibangun diuji dengan menggunakan 17 alternatif dan 3 kriteria yang terdiri dari 1 kriteria cost dan 2 benefit. Eksperimen yang dilakukan berhasil memberikan perankingan yang berbeda terhadap 15 alternatif dan hanya 2 alternatif dengan ranking yang sama yaitu pada ranking ke-5 dan ke-6 karena skorkeduanya sama pada setiap kriteria.

Oleh Nandik Sesnika, Desi Andreswari, Rusdi Efendi (2016) jurusan Teknik Informatika, Universitas Bengkulu dengan judul Aplikasi Sistem Pendukung Keputusan Pemilihan Gedung Serba Guna Di Kota Bengkulu Dengan Menggunakan Metode Smart Berbasis Android. Pada penelitian ini dibangun sebuah aplikasi yang digunakan untuk memilih dan mencari informasi mengenai gedung serba guna di Kota Bengkulu. Aplikasi ini dibangun berbasis Android dengan menggunakan Bengkulu dengan menggunakan sebuah metode yang bernama Simple Multi Attribute Rating Technique (SMART), dan dibangun dengan menggunakan bahasa pemrograman JAVA dengan IDE ECLIPSE JUNO. Analisis perancangan sistem ini menggunakan Unified Modeling Language (UML). Dapat disimpulkan bahwa aplikasi ini dapat membantu para pengguna dalam memilih gedung serba guana yang ada di Kota Bengkulu dengan menggunakan metode SMART. Hasil pencarian yang ditampilkan berupa daftar nama gedung serba guna yang didalamnya terdapat berbagai informasi mengenai gedung-gedung serba guna tersebut.

Ringkasan tinjauan literatur dari beberapa penelitian yang berkaitan dengan penelitian ini yaitu ditunjukkan pada Tabel 2.1.

Tabel 2.1 Tinjauan Pustaka

No Literatur	Penulis	Tahun	Judul Penelitian
01	Titin Kristina	2018	Sistem Pendukung Keputusan Dengan Menggunakan Metode TOPSIS Untuk Pemilihan Lokasi Pendirian Grosir Pulsa
02	Avriana Indarwasti, Barizana S.A, Prottasof G.K	2017	Sistem Pendukung Keputusan Pemilihan Tempat Kuliner Di Depok Dengan TOPSIS
03	Muhamad Aries Permana, Anggi Srimurdianti Sukamto, Tursina	2017	Rancang Bangun Aplikasi Sistem Pendukung Keputusan Pemilihan Gedung Lokasi Pernikahan Menggunakan Metode Promethee (Studi Kasus: Pontianak)
04	Putri Alit Widyastuti Santiary, Putu Indah Ciptayani, Ni G. A. Harry	2018	Sistem Pendukung Keputusan Penentuan Lokasi Wisata Dengan Metode TOPSIS
05	Nandik Sesnika, Desi Andreswari, Rusdi Efendi	2016	Aplikasi Sistem Pendukung Keputusan Pemilihan Gedung Serba Guna Di Kota Bengkulu Dengan Menggunakan Metode Smart Berbasis Android

2.2 Gedung Serbaguna

Gedung serbaguna adalah bangunan yang berfungsi untuk menampung kegiatan yang berbeda-beda, yang mana masing – masing kegiatan memiliki kaitan yang erat dan saling melengkapi satu sama lain serta memenuhi kriteria yang ada dalam konteks tertentu (berkaitan dengan fungsi utama bangunan tersebut). Merencanakan gedung serbaguna bukan hanya menggabungkan beberapa kegiatan yang berbeda dalam suatu bangunan yang fleksibel, tetapi bagaimana membangun suatu lingkungan yang memiliki integrasi fisik maupun fungsi dari komponen – komponen yang ada sehingga saling melengkapi dan menunjang, serta menciptakan lingkungan baru yang harmonis (Mayer, 1983).

2.3 Sistem

Pada dasarnya sistem merupakan suatu proses yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai suatu tujuan. Menurut Laudon dan Laudon (2010), sistem merupakan komponen yang saling bekerjasama untuk mengumpulkan, mengolah, menyimpan dan menyebarkan informasi untuk mendukung pengambilan keputusan, koordinasi, pengendalian, analisis masalah dan visualisasi dalam sebuah organisasi.

Sedangkan Stain dan Reynolds (2010) mendefinikan sistem sebagai seperangkat elemen atau komponen yang saling terkait yang dikumpulkan (input), memanipulasi (process), menyimpan, dan menyebarkan (output) data dan , dan memberikan reaksi korektif (feedback) untuk memenuhi tujuan.

2.4 Sistem Pendukung Keputusan

Sistem pendukung keputusan (SPK) adalah sistem yang mampu memberikan kemampuan pemecahan masalah maupun kemampuan pengkomunikasian untuk masalah dengan kondisi semi terstruktur dan tak terstruktur. Sistem ini digunakan untuk membantu pengambilan keputusan dalamsituasi semi terstruktur dan situasi yang tidak terstruktur, di mana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat (Turban E, 2005).

Pada dasarnya sistem pendukung keputusan merupakan pengembangan lebih lanjut dari sistem informasi manajemen terkomputerisasi yang dirancang sedemikian rupa sehingga bersifat interaktifdengan pemakainya. Sifat interaktif dimaksudkan untuk memudahkan integrasi antara berbagai komponen dalam proses pengambilan keputusan seperti prosedur, kebijakan, teknik analisis, serta pengalaman dan wawasan manajerial guna membentuk suatu kerangka keputusan bersifat fleksibel. Konsep Sistem Pendukung Keputusan (SPK) *Decision Support Sistem* (DSS) pertama kali diungkapkan pada awal tahun 1970-an oleh Michael S. Scott Morton dengan istilah *Management Decision Sistem*. Sistem tersebut adalah suatu sistem yang berbasis komputer yang ditujukan untuk membantu pengambil keputusan dengan memanfaatkan data dan model tertentu untuk memecahkan

berbagai persoalan yang tidak terstruktur. Menurut Kadarsah, S, (1998) yang dikutip oleh (Adiwisanghagni, 2015).

2.5 Metode TOPSIS

Metode TOPSIS (*Technique For Others Reference by Similarity to Ideal Solution*) dipilih karena konsepnya sederhana dan mudah dipahami, komputasinya efisien dan memiliki kemampuan untuk mengukur kinerja relatif dari alternatif-alternatif keputusan dalam bentuk matematis yang sederhana. Metode ini diharapkan dapat membantu pemilihan lokasi yang strategis dan yang sesuai dengan yang diharapkan. Keunggulan metode TOPSIS dalam pengambilan keputusan masalah yang kompleks atau mudah digunakan dan dapat memperhitungkan semua jenis kriteria (subyektif dan obyektif), serta proses perhitungan yang sederhana, mudah dipahami dan bobot penting dapat dimasukkan dengan mudah (Adiwisanghagni, 2015).

Langkah-langkah penyelesaian masalah dengan metode TOPSIS (Indrawasti, 2017):

1. Membuat matriks keputusan yang ternormalisasi,
2. Membuat matriks keputusan yang ternormalisasi terbobot,
3. Menentukan matriks solusi ideal positif & matriks solusi ideal negatif,
4. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif & matriks solusi ideal negatif,
5. Menentukan nilai preferensi untuk setiap alternatif.

2.5.1 Tahapan Dalam Metode TOPSIS

Menurut (Adiwisanghagni, 2015)., beberapa langkah dalam penyelesaian metode TOPSIS :

1. Membuat *matriks* keputusan yang ternormalisasi *Elemen rij* hasil dari normalisasi *decision matrix* R dengan metode *Eucliciden length of a vector* adalah :

$$r\ ij = \frac{Xij}{\sqrt{\sum_{i=1}^{m} x2_{ij}}}$$

Dimana :

rij = hasil dari normalisasi matriks keputusan R

i = 1,2,3,...,m;

j = 1,2,3,...,n;

x_{ij} = elemen dari matriks keputusan, i = 1,2,3, ..., m, j = 1, 2, 3, ..., n.

2. Membuat *matriks* keputusan yang ternormalisasi terbobot dengan bobot

W = (w1,w2,….), maka normalisasi bobot *matriks* V adalah :

$$\begin{bmatrix} w_{11}r_{11} & \cdots & w_{1n}r_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1}r_{m1} & \cdots & w_{nm}r_{nm} \end{bmatrix}$$

3. Menentukan matriks solusi ideal positif dan matriks solusi ideal negatif

Solusi ideal positif dinotasikan dengan A+ dan solusi dan solusi ideal negatif dinotasikan dengan A sebagai berikut :

Menentukan solusi ideal (+) dan (-)

$$A^{+} = \{(\max v_{ij} \mid j \in J), (\min v_{ij} \mid j \in J^{2}), i = 1, 2, 3, \ldots, m\}$$

$$= \{v_1^{-}, v_2^{-}, v_3^{-}, \ldots, v_n^{-}\}|$$

$$A^{-} = \{(\max v_{ij} \mid j \in J), (\min v_{ij} \mid j \in J^{2}), i = 1, 2, 3, \ldots, m\}$$

$$= \{v_1^{-}, v_2^{-}, v_3^{-}, \ldots, v_n^{-}\}$$

Dimana :

***v** ij* = elemen matriks *V* baris ke-*i* dan kolom ke-*j*

J = {j=1,2,3,...,n dan j berhubungan dengan *benefit criteria*}

J ' = {j=1,2,3,...,n dan j berhubungan dengan*cost criteria*}

4. Menentukan Separasi

Separation measure merupakan pengukuran jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan negatif.

Perhitungan jarak untuk solusi ideal positif :

$$S^{+} = \sqrt{\sum_{j}^{n} {}_{1} \ (v_{ij-v_j^{-}})^{2}}$$, dengan I = 1, 2, 3, ... , m

Dimana :

J = {j=1,2,3,...,n dan j merupakan *benefit criteria*}

J' = {j=1,2,3,...,n dan j merupakan *cost criteria*}

Perhitungan Jarak untuk solusi ideal negatif :

$$S^{-} = \sqrt{\sum_{j}^{n} {}_{1} \ (v_{ij-v_j^{-}})^{2}}$$, dengan I = 1, 2, 3, ... , m

Dimana :

J = {j=1,2,3,...,n dan j merupakan *benefit criteria*}

J' = {j=1,2,3,...,n dan j merupakan *cost criteria*}

5. Menentukan kedekatan *relatif* terhadap solusi ideal yang akan diambil kedekatan *relatif* dari alternatif A+ dengan solusi ideal A- direpresentasikan dengan :

$$C_i^{-} = \frac{S_i^{-}}{(S_i^{-} + S_i^{+})}, 0 \leq C_i^{-} \leq 1,$$ dengan I = 1, 2, 3, ... , m

6. Menentukan *ranking alternatif*

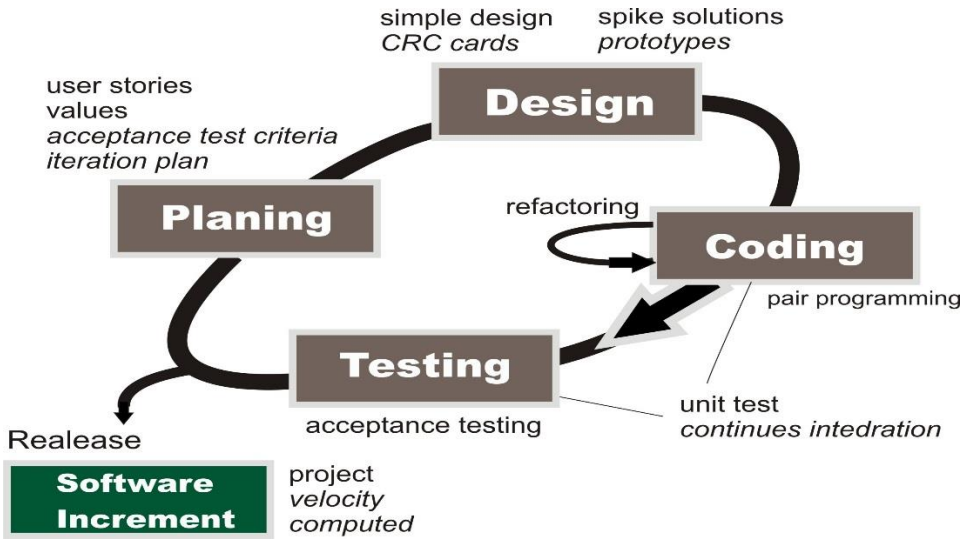
Alternatif dapat diranking berdasarkan urutan *Ci**. Maka dari itu, alternatif terbaik adalah salah satu yang berjarak terpendek terhadap solusi ideal dan berjarak terjauh dengan solusi ideal negatif.

2.6 Metode Pengembangan Sistem

Pengembangan sistem berarti menyusun sistem baru untuk mengganti sistem lama secara keseluruhan atau memperbaiki bagian-bagian tertentu dalam sistem lama. Metode yang digunakan dalam pengembangan sistem yaitu dengan siklus klasik atau air terjun dengan tahapan - tahapan yang terdiri dari survei sistem, analisis sistem, pembuatan sistem, implementasi sistem, pengujian dan pemeliharaan sistem. Dalam metode air terjun setiap tahun harus diselesaikan

terlebih dahulu secara penuh sebelum diteruskan ke tahap berikutnya untuk menghindari pengulangan tahapan.

Extreme Programming (XP) adalah metodologi pengembangan perangkat lunak yang ditujukan untuk meningkatkan kualitas perangkat lunak dan tanggap terhadap perubahan kebutuhan pelanggan. Jenis pengembangan perangkat lunak semacam ini dimaksudkan untuk meningkatkan produktivitas dan memperkenalkan pos pemeriksaan dimana persyaratan pelanggan baru dapat diadopsi. Tahapan - tahapan dari *Extreme Programming* terdiri dari *planning* seperti memahami kriteria pengguna dan perencanaan pengembangan, *designing* seperti perancangan *prototype* dan tampilan, *coding* termasuk pengintegrasian, dan yang terakhir adalah *testing* (Pressman, Roger, S, 2012).



Gambar 2.1 Model *Extreme Programming (XP)***,** (Pressman, Roger, S, 2012)

2.6.1 Proses *Extreme Programming* (XP)

Proses *Extreme Programming* (XP) menurut (Pressman, Roger, S, 2012) :

1. *Planning* : Tahap *planning* dimulai dengan membuat *user stories* yang menggambarkan output, fitur, dan fungsi - fungsi dari *software* yang akan dibuat. *User stories* tersebut kemudian diberikan bobot seperti prioritas dan dikelompokkan untuk selanjutnya dilakukan proses *delivery* secara *incremental*.

2. *Design* : *Design* di *Extreme Programming* mengikuti prinsip *Keep It Simple* (KIS). Untuk *design* yang sulit, *Extreme Programming* akan menggunaan *Spike Solution* dimana pembuatan design dibuat langsung ke tujuannya. *Extreme Programming* juga mendukung adanya *refactoring* dimana *software system* diubah sedemikian rupa dengan cara mengubah stuktur kode dan menyederhanakannya namun hasil dari kode tidak berubah.
3. *Coding* : Proses *coding* pada *Exterime Programming* diawali dengan membangun serangkaian unit test. Setelah itu pengembangan akan berfokus untuk mengimplementasikannya. Dalam *Exterime Programming* diperkenalkan istilah Pair Programming dimana proses penulisan program dilakukan secara berpasangan. Dua orang *Programmer* saling bekerjasama di satu komputer untuk menulis program. Dengan melakukan ini akan didapat *real-time problem solving* dan *real-time quality assurance.*
4. *Testing* : Tahap ini dilakukan pengujian kode pada unit test. Dalam *Extreme Programming,* diperkenalkan *Extreme Programming acceptance test* atau biasa disebut *customer test*. Tes ini dilakukan oleh customer yang berfokus kepada fitur dan fungsi sistem secara keseluruhan. *Acceptance test* ini berasal dari *user stories* yang telah diimplementasikan.

2.7 Alat Perancangan Sistem

Alat perancangan sistem merupakan serangkaian model dan representasi teknis dari sistem yang diusulkan. Sasaran utama dari alat perancangan sebagai berikut :

1. Menggambarkan apa yang dibutuhkan oleh perusahaan.
2. Membangun dasar bagi pembuatan desain perangkat lunak.
3. Membatasi serangkaian persyaratan yang dapat divalidasi begitu perangkat lunak dibangun.

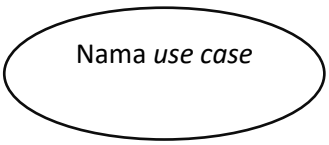
2.7.1. *Unified Modeling Languege (UML)*

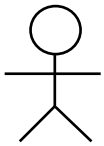
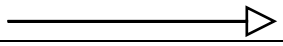
Menurut Rosa dan Shalahuddin (2013), *Unified Modelling Languange* (UML) merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. Pada perkembangan teknik pemograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang di bangun dengan menggunakan teknik pemograman berorientasi objek yaitu *Unified Modeling Language* (UML) dalam UML terdiri dari 13 macam diagram yang dikelompokan dalam 4 kategori sebagai berikut :

2.7.1.1. *Use Case Diagram*

Merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. *Usecase* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.secara kasar *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Tabel 2.2 Simbol-simbol *Use Case Diagram*, (Rosa A.S, 2014)

No.	Simbol	Keterangan
1.	*Use Case*  	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antara unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *Use Case.*

2.	Aktor / *actor* 	Orang, aktor, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
3.	Ekstensi / *extend* <<*extend*>>	Relasi *Use Case* tambahan kesebuah *Use Case* Dimana *Use Case* yang ditambahkan dapat berdiri sendiri walau tanpa *Use Case* tambahan itu; mirip dengan prinsip *inheritance* pada *OOP*; biasanya *Use Case* tambahan memiliki nama depan yang sama dengan *Use Case* yang ditambahkan.
4.	*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.
5.	Generalisasi 	Relasi antara kelas dengan makna generalisasi-generalisasi (umum khusus).
6.	Asosiasi / *Association*	Komunikasi antara aktor dan *Use Case* yang berpartisipasi pada *Use Case* atau *Use Case* memiliki interaksi dengan aktor.

2.7.1.2. *Class Diagram*

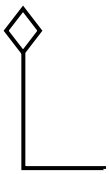
Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Diagram kelas dibuat agar pembuatan perogram

membuat kelas-kelas sesuai rancangan didalam diagram kelas antara dokumentasi perancangan dan perangkat lunak singkron. Berikut ini adalah simbol-simbol yang ada pada Class Diagram.

Tabel 2.3 Simbol-simbol *Class Diagram*, (Rosa A.S, 2014)

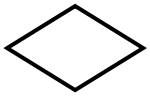
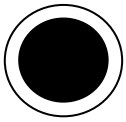
No.	**Simbol**	**Keterangan**
1.	Kelas ClassName - memberName1 : int - memberName2 : int	Kelas pada struktur sistem.
2.	Antarmuka/*Interface*	Sama dengan konsep interface dalam pemrograman berorientasi objek.
3	Asosiasi/*Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity.
4	Asosiasi berarah/ *directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity.
5	Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus).
6	Kebergantungan/*dependensi*	Relasi antar kelas dengan makna kebergantungan antar kelas.

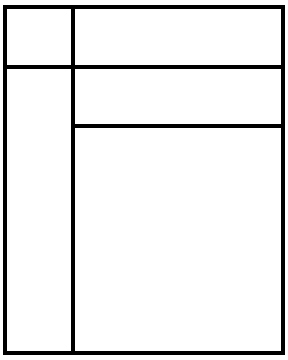
7	Agrgasi/*aggregation* 	Relasi antar kelas dengan makna kebergantungan antar kelas relasi antar kelas dengan makna semua-bagian (*whole-part*)

2.7.1.3. *Activity Diagram*

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Tabel 2.4 Simbol-simbol *Activity Diagram*, (Rosa A.S, 2014)

No.	Simbol	Keterangan
1.	Status Awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal. (Mulai/*Start*)
2.	Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan/*desicion* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.	Penggabungan/*join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.	Status Akhir 	Status akhir yang dilakukan oleh sistem, sebuah diagram aktivitas memiliki sebuah status akhir. (Selesai/*Finish*)

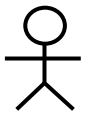
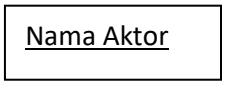
6.	*Swimlane* 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.7.1.4. *Sequence Diagram*

Sequence Diagram menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek atau *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sequence maka harus diketahui objek-objek yang terlibat dalam sebuah usecase berserta metode-metode yang dimiliki kelas yang di instansiasi menjadi objek itu.

1. Sistem menghasilkan produk yang tidak akurat.
2. Sistem menghasilkan produk yang tidak konsisten.
3. Sistem menghasilkan produk yang tidak dipercaya.
4. Sistem tidak mudah dipelajari.
5. Sistem tidak mudah digunakan.

Tabel 2.5 Simbol-simbol *Sequence Diagram,* (Rosa A.S, 2014)

No.	Simbol	Keterangan
1.	Aktor  atau 	Orang, aktor, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

2.	Garis hidup/*lifeline*	Menyatakan kehidupan suatu objek.
3.	Objek Nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan.
4.	Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semuanya yang terhubungdengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya 1 : login() 2 : cekStatusLogin() 3 : open() Maka cekStatusLogin() dan open() dilakukan didalam metode login(). Aktor tidak memiliki waktu aktif.
3	Pesan tipe *create* << create>>	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
4	Pesan tipe *call* 1 : nama_metode()	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri, 1 : nama_metode() 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
5	Pesan tipe *send* 1 : masukkan ⟶	Menyatakan bahwa suatu objek mengirimkan data/masukkan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.
8.	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.
9.	Pesan tipe *destroy*	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaliknya jika ada create maka ada *destroy*.

2.7.2. *Android*

Menurut Teguh Afrianto (2011) android merupakan perangkat bergerak pada sistem operasi untuk telepon seluler yang berbasis linux.

Menurut Hermawan (2011) android merupakan OS (*Operating System*) *mobile* yang tumbuh ditengah OS lainnya yang berkembang dimasa sekarang ini. OS lainnya seperti *Windows Mobile, i-Phone OS,* Symbian dan masih banyak lagi. Akan tetapi, OS yang ada ini berjalan dengan memprioritaskan aplikasi inti yang dibangun sendiri tanpa melihat potensi yang cukup besar dari aplikasi pihak ketiga

Menurut Busran (2015) Android adalah suatu sistem operasi yang didesain sebagai platform *open source* untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware*, dan aplikasi. Android menyediakan platform yang terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Android juga menyediakan semua tools dan framework untuk mengembakan aplikasi dengan mudah dan cepat. Dengan adanya Android SDK (*Software Development Kit*) pengembang aplikasi dapat memulai pembuatan aplikasi pada platform android menggunakan bahasa pemrograman Java.

Sedangkan menurut Nazruddin Safaat (2012), android merupakan salah satu platform sistem operasi yang digemari masyarakat karena sifatnya yang *open source* sehingga memungkinkan pengguna untuk melakukan pengembangan, arsitektur android terdiri dari bagian – bagian seperti berikut :

1. *Applications* dan *Widgets*: layer (lapisan) dimana pengguna hanya berhubungan dengan aplikasi saja.
2. *Applications Framework*: lapisan dimana para pengembang melakukan pembuatan aplikasi yang akan dijalankan di sistem operasi Android dengan komponen – komponennya meliputi *views, contents provider, resource manager, notification manager, activity manager.*
3. *Libraries*: lapisan dimana fitur – fitur android ditempatkan yang berada diatas kernel meliputi library C/C++ inti seperti Libc dan SSL.
4. *Android Run Time*: lapisan yang membuat aplikasi android dapat dijalankan, dimana dalam prosesnya menggunakan implementasi Linux yang terbagi menjadi dua bagian yaitu *Core Libraries* dan *Dalvix virtual Machine*.

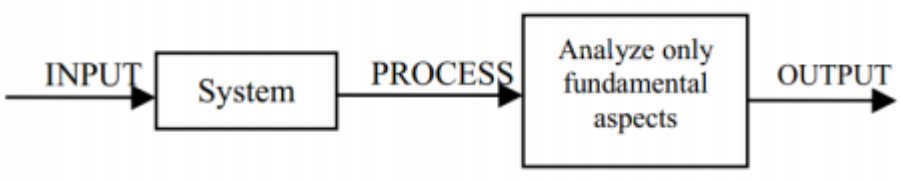
Linux Kernel: layer yang berisi file – file sistem untuk mengatur *proccesing, memory, resource, driver* dan sistem operasi android lainnya.

2.7.3. Pengujian *Black-Box*

Menurut (A.S Rosa dan Shalahuddin,M , 2013) mengatakan bahwa pengujian *Black-Box* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi – fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian kotak hitam dilakukan dengan membuat kasus uji yang mencoba semua fungsi dengan memakai perangkat lunak, apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian *black-box* dengan kasus benar dan kasus salah.

Menurut (Permana, Sukamto and Tursina, 2016) metode *Black Box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. *Black Box* dapat menemukan kesalahan dalam kategori berikut:

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan interface
3. Kesalahan dalam strutur data atau akses basisdata eksternal
4. Inisialisasi dan kesalahan terminasi
5. Validitas fungsional
6. Kesensitifan sistem terhadap nilai input tertentu
7. Batasan dari suatu data



Gambar 2.2 Sistem Kerja dan Teknik Pengujian *Black Box*