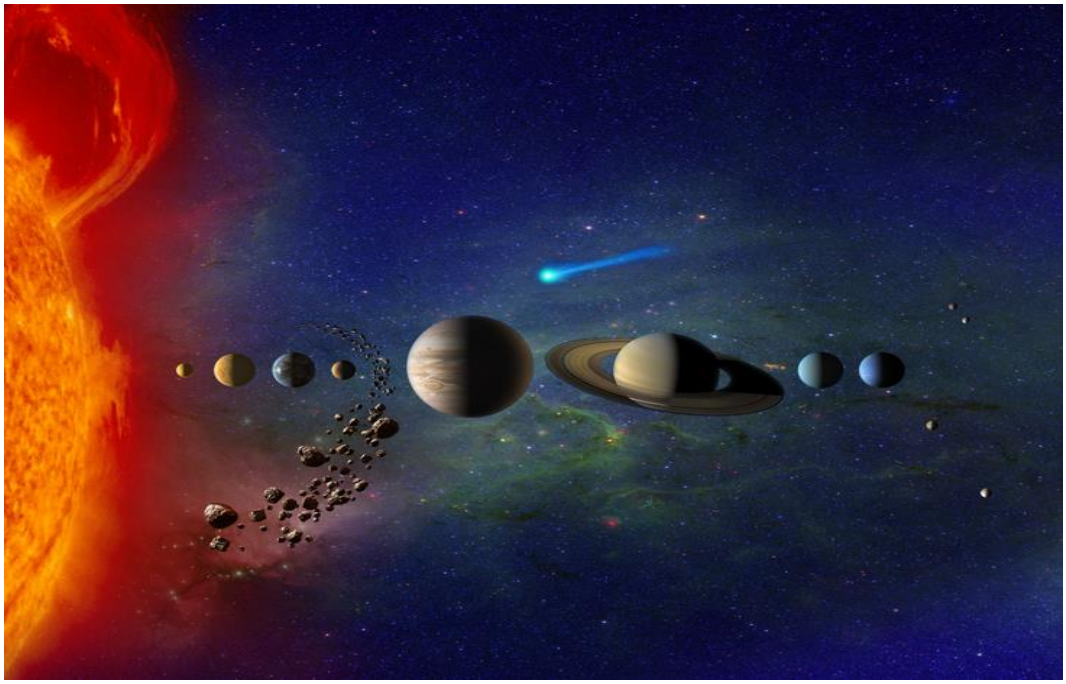
BAB II
LANDASAN TEORI

2.1. Sistem Tata Surya

Tata Surya merupakan sebuah sebuah sistem yang terdiri dari Matahari, planet, planet-kerdil, komet, asteroid dan benda-benda angkasa kecil lainnya. Matahari merupakan pusat dari Tata Surya di mana anggota Tata Surya yang lain beredar mengelilingi Matahari.

Di dalam sistem Tata Surya terdapat delapan planet, lima planet kerdil, satu bintang yang biasa kita sebut dengan matahari, 3.319 komet, dan 670.452 asteroid [10]. Benda-benda langit tersebut beredar mengelilingi Matahari secara konsentris pada lintasannya masing-masing. Anggota-anggota dalam sistem Tata Surya ditunjukkan seperti gambar 2.1.



Gambar 2. 1 Anggota Sistem Tata Surya [11]

Benda-benda langit tersebut beredar mengelilingi Matahari secara konsentris pada lintasan masing-masing. IAU (*International Astronomical Union*) secara umum mengelompokkan benda angkasa yang mengelilingi Matahari menjadi tiga [12], yaitu:

1. **Planet**

   Sebuah benda langit dikatakan planet jika memenuhi kriteria sebagai berikut:

   a. Mengorbit Matahari.
   b. Memiliki massa yang cukup untuk membuat gravitasinya sendiri sehingga bentuk fisiknya cenderung bulat.
   c. Orbitnya bersih dari keberadaan benda angkasa lain

2. **Planet Kerdil**

   Planet-Kerdil Sebuah benda langit dikatakan sebagai planet-kerdil jika:

   a. Mengorbit Matahari.
   b. Memiliki massa yang cukup untuk membuat gravitasinya sendiri sehingga bentuk fisiknya cenderung bulat.
   c. Orbitnya belum bersih dari keberadaan benda angkasa lain.
   d. Bukan merupakan satelit.

3. **Benda-benda Tata Surya Kecil (*Small Solar System Bodies*)**

   Benda-benda Tata Surya Kecil tersebut di antaranya adalah komet, asteroid, objek-objek trans-neptunian, serta benda-benda kecil lainnya.

2.1.1. Anggota Tata Surya

Anggota tata surya terdiri dari matahari, planet utama, planet kerdil, asteroid, komet, satelit, dan benda angkasa lainnya.

2.1.1.1. Matahari

Matahari merupakan sebuah bintang yang jaraknya paling dekat ke Bumi. Jarak rata-rata Bumi ke Matahari adalah 149,6 juta Km atau 1 Satuan Astronomi dan biasanya disingkat dengan nama AU (*Astronomical Unit*) dan ditetapkan sebagai skala untuk mengukur jarak di seluruh tata surya. Matahari adalah sebuah bola besar yang terdiri dari gas yang terionisasi. Koneksi dan interaksi antara matahari dan bumi mempengaruhi musim, arus laut, cuaca dan iklim.

Matahari memiliki volume satu juta kali dari diameter bumi serta terbagi menjadi enam bagian (inti, zona radiatif, zona konvektif, fotosfer, kromosfer, bagian terluar atau *corona*) [13].

2.1.1.2. Planet

Planet-planet yang berada dalam sistem Tata Surya adalah : Merkurius, Venus, Bumi, Mars, Jupiter, Saturnus, Uranus, dan Neptunus. Sejak tahun 2006, Pluto tidak dikategorikan lagi sebagai planet karena Pluto tidak memiliki orbit yang bersih dari benda angkasa lainnya. Pluto memiliki orbit yang memotong orbit Neptunus sehingga dianggap orbit Pluto belum bersih dari benda angkasa lain.

a. Merkurius

Merupakan planet yang paling dekat ke Matahari dengan jarak 0,39 SA. Karena planet Merkurius jaraknya paling dekat ke Matahari, maka suhu pada siang hari di Merkurius mencapai 4270 °C, sedangkan pada malam hari suhunya menjadi sangat rendah yaitu mencapai –1700 °C. Merkurius mempunyai eksentrisitas yang besar yaitu 0,206 akibatnya jarak antara Merkurius dan Matahari bervariasi dengan cukup besar pula. Perbedaan jarak terjauh ke Matahari (*aphelium*) dengan jarak terdekat ke Matahari (*perihelium*) adalah sebesar 22 juta Km. Jarak aphelium planet Merkurius adalah 57,9 juta km. Merkurius tidak memiliki atmosfir oleh karena hal tersebut langit Merkurius berwarna hitam. Kerapatan atau densitasnya 5,43 gr/cm3.

b. Venus

Planet Venus lebih dikenal sebagai Bintang Kejora atau Bintang Senja. Eksentrisitas planet Venus adalah 0,007, sehingga orbit planet Venus mendekati bentuk lingkaran. Jarak Venus ke Matahari 0,72 SA, sehingga di Venus suhunya sangat panas dapat mencapai 4800 °C. Tingginya suhu di planet Venus diakibatkan adanya efek rumah kaca. Kerapatan atau densitas Venus adalah 5,24 gr/cm3.

c. Bumi

Sampai saat ini Bumi merupakan satu-satunya planet yang mempunyai kehidupan. Hal tersebut dimungkinkan karena Bumi diselubungi oleh atmosfirnya sehingga perbedaan suhu pada siang dan malam tidak terlalu besar. Bumi mengorbit Matahari sebagai bintang pusatnya dengan eksentrisitas 0,017, sehingga orbitnya hampir membentuk lingkaran. Jarak rata-rata Bumi ke Matahari adalah 1 Satuan Astronomi atau 150 juta kilometer. Kala revolusi Bumi adalah 365,3 hari, sedangkan kala rotasinya adalah 23 jam 56 menit.

Kerapatan atau densitas Bumi adalah 5,52 gram/cm3, Bumi merupakan benda terpadat dalam sistem Tata Surya. Bumi mempunyai sebuah satelit yaitu Bulan.

d. Mars

Jarak rata-rata planet Mars ke Matahari adalah 1,52 SA atau 228 juta kilometer dengan eksentrisitas 0,093. Mars berputar mengelilingi Matahari dengan kala revolusi 687 hari. Mars mempunyai dua buah satelit yaitu Phobos dan Deimos.

e. Jupiter

Jarak rata-rata planet Jupiter ke Matahari adalah 5,2 SA. Jupiter mempunyai eksentrisitas 0,048 dengan kala revolusi 11,86 tahun. Jupiter memiliki 67 buah satelit. Empat buah satelitnya yang berukuran besar bernama IO, Europa, Ganymede, dan Callisto. Yupiter merupakan planet terbesar dalam sistem tata surya; mempunyai kala rotasi 9 jam 50 menit; artinya Yupiter berotasi dengan sangat cepat.

f. Saturnus

Jarak rata-rata Saturnus ke Matahari adalah 9,5 SA. Saturnus mempunyai eksentrisitas 0,056 dengan kala revolusi 29,5 tahun. Saturnus dihiasi oleh gelang dan cincin yang indah, mempunyai 9 buah satelit yaitu Mimas, Enceladus, Tethys, Dione, Rhea, Titan, Hyperion, Lapetus, dan Phoebe.

g. Uranus

Jarak rata-rata planet Uranus ke Matahari adalah 19,2 SA. Uranus mempunyai eksentrisitas 0,047 dengan kala revolusi 84 tahun. Uranus mempunyai cincin dan mempunyai 5 buah satelit yaitu Miranda, Ariel, Umbriel, Titania, dan Oberion.

h. Neptunus

Jarak rata-rata planet Neptunus ke Matahari adalah 30,07 SA. Neptunus mempunyai eksentrisitas 0,009 dengan kala revolusi 164,8 tahun. Neptunus mempunyai dua buah satelit yaitu Triton dan Nereid.

2.1.1.3. Planet Kerdil

Planet-kerdil (*Dwarf Planet*) merupakan kategori baru dalam keanggotaan Tata Surya, berdasarkan resolusi IAU tahun 2006. Planet kerdil yang berada dalam sistem Tata Surya adalah : Ceres, Makemake, Haumea, Eris, dan Pluto.

a. Ceres

Ditemukan pada tahun 1801 dan pertama kali dikenalkan sebagai sebuah planet, lalu dianggap sebagai sebuah asteroid, namun saat ini disebut sebuah planet kerdil dengan nama Ceres. Ceres merupakan satu-satunya planet kerdil yang berada di bagian dalam dari Tata Surya dan merupakan planet kerdil terkecil.

b. Makemake

Ditemukan pada Maret 2005, Makemake merupakan planet kerdil kedua terjauh dari matahari dan merupakan planet kerdil ketiga terbesar.

c. Haumea

Ditemukan pada Desember 2004, Haumea merupakan planet kerdil ketiga terdekat dari matahari dan merupakan planet dengan rotasi tercepat yaitu 6,1 kali lebih cepat dari bumi.

d. Eris

Ditemukan pada Januari 2005, Eris merupakan planet kerdil terjauh dari matahari dan merupakan planet kerdil terbesar.

e. Pluto

Pluto merupakan objek terbesar di *kuiper belt* dan planet kerdil terbesar kedua pada Tata Surya.

2.1.1.4. Satelit

Satelit adalah benda langit pengiring planet. Satelit senantiasa mengiringi dan berputar terhadap planet pusatnya. Berdasarkan cara terbentuknya satelit dapat dibedakan menjadi 2 bagian, yaitu :

a. Satelit Alam

Satelit alam adalah satelit yang terbentuk karena adanya peristiwa alam bersamaan dengan terbentuknya planet. Contoh: Bulan, sebagai satelit alam Bumi; Titan, sebagai satelit alam Saturnus.

b. Satelit Buatan

Satelit buatan adalah satelit yang dibuat oleh manusia yang digunakan untuk tujuan tertentu. Contoh: Satelit cuaca, satelit komunikasi, satelit mata-mata, dan sebagainya.

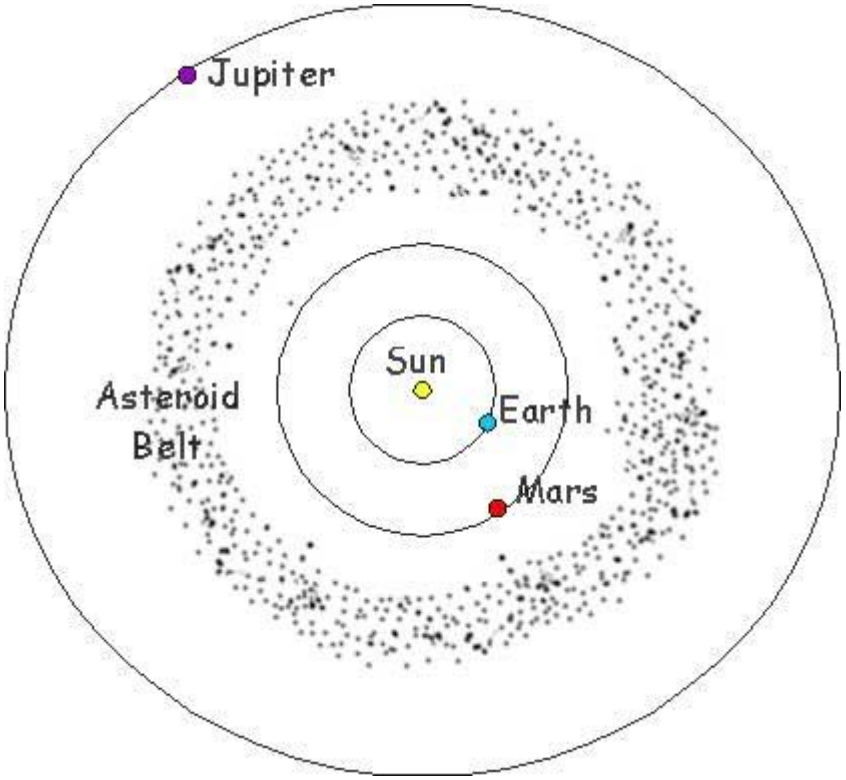
Pada umumnya planet-planet dalam sistem tata surya mempunyai beberapa satelit yang senantiasa mengiringinya. Hanya planet Merkurius dan planet Venus yang tidak memiliki satelit. Jumlah masing-masing satelit untuk setiap planet ditunjukkan pada tabel 2.1 di bawah ini:

Tabel 2. 1 Jumlah Satelit Alam tiap Planet [14]

No.	Nama Planet	Jumlah Satelit Alam
1	Merkurius	0
2	Venus	0
3	Bumi	1
4	Mars	2
5	Jupiter	67
6	Saturnus	62
7	Uranus	27
8	Neptunus	14
9	Ceres	0
10	Makemake	0
11	Haumea	2
12	Eris	1
13	Pluto	5

2.1.1.5. Asteroid

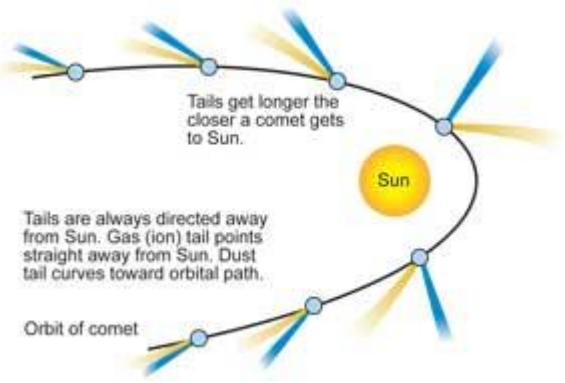
Asteroid dinamakan juga planet minor atau planetoid. Asteroid mengisi ruangan yang berada diantara Mars dan Jupiter. Di dalam sistem Tata Surya ditaksir terdapat 100.000 buah planetoid yang ukurannya antara 2–750 Km2. Asteroid-asteroid tersebut senantiasa berputar diantara planet Mars dan planet Jupiter membentuk sabuk asteroid. Sabuk Asteroid ditunjukkan seperti pada gambar 2.2.



Gambar 2. 2 Sabuk Asteroid [15]

2.1.1.6. Komet

Dinamakan juga “Bintang berekor“, merupakan benda langit yang garis edarnya/orbitnya sangat lonjong sehingga jaraknya ke Matahari kadang-kadang jauh sekali tetapi suatu saat dapat dekat sekali. Wujud komet tersusun dari kristal-kristal es yang rapuh sehingga mudah terlepas dari badannya. Bagian yang terlepas inilah yang membentuk semburan ketika sebuah komet melintas di dekat Matahari. Semburan tersebut nampak seperti ekor yang selalu menjauhi Matahari sebab mendapat tekanan dari Matahari. Karena orbit komet tidak seperti orbit planet maka komet akan terlihat di bumi jika komet tersebut berada dekat dengan Matahari. Oleh karena itu ada komet yang mendekati Bumi setiap 3 atau 4 tahun sekali; tetapi ada juga yang sampai 76 tahun sekali yaitu Komet Halley. Bentuk lintasan komet ditunjukkan seperti gambar 2.3.



Gambar 2. 3 Bentuk Lintasan Komet [16]

2.2. Perhitungan Posisi Benda Angkasa

Sebuah benda angkasa biasanya mengorbit matahari dalam orbit elips. Gangguan dari planet lain menyebabkan penyimpangan kecil dari orbit elips ini, tetapi orbit elips tanpa gangguan dapat digunakan sebagai pendekatan pertama, dan kadang-kadang digunakan sebagai pendekatan akhir. Jika jarak dari Matahari ke planet selalu sama, maka planet mengikuti orbit lingkaran. Namun pada kenyataannya tidak ada planet yang seperti ini. [17]

2.2.1. Istilah Dalam Perhitungan

Pada saat dilakukan perhitungan posisi planet secara real terdapat beberapa istilah yang digunakan, yaitu:

1. *Perihelion* adalah titik orbit terdekat planet dari matahari.
2. *Aphelion* adalah titik orbit terjauh planet dari matahari.
3. *Perigee* adalah titik terjauh orbit bulan dari bumi.
4. *Apogee* adalah titik terdekat orbit bulan dari bumi.
5. *Ecliptic* adalah bidang orbit dari bumi.
6. *Right Ascension* adalah koordinat langit yang mengacu pada garis lintang bumi.
7. *Declination* adalah koordinat langit yang mengacu pada garis bujur bumi.
8. *Longitude* (Garis Bujur)
9. *Latitude* (Garis Lintang)
10. *Heliosentric* adalah posisi relatif terhadap matahari.
11. *Geosentric* adalah posisi realtif terhadap bumi.

12. *Topocentric* adalah posisi terhadap pengamat di permukaan bumi.
13. *Eccentricity* adalah angka yang mendefinisikan seberapa jauh orbit sebuah planet menyimpang.
14. *Inclination* adalah kemiringan orbit relatif terhadap orbit yang seharusnya.

Simbol-simbol yang dipakai pada perhitungan dapat dilihat pada tabel 2.2.

Tabel 2. 2 Tabel Simbol Perhitungan

Simbol	Keterangan
a	Jarak rata-rata dalam satuan astronomy
e	Penyimpangan jalur orbit
T	Waktu pada saat *perihelion*
I	*Inclination*
N	*Longitude of Ascending Node*
W	Sudut dari *Ascending Node* terhadap *Perihelion*
q	Jarak *perihelion*
Q	Jarak *aphelion*
P	Waktu orbit
n	Pergerakan perhari
t	Perhitungan hari
M	*Mean Anomaly*
L	*Mean Longitude*
E	*Eccentric Anomaly*
v	*True Anomaly*
r	Jarak *heliocentric*
x,y,z	Koordinat persegi panajang

2.2.2. *Julian Date*

Julian Date (JD) adalah sistem pengukuran waktu yang digunakan untuk penelitian ilmiah oleh komunitas astronomi. JD adalah interval waktu pada satuan hari dan fraksi sehari sejak 1 Januari 4713SM waktu Greenwich siang hari.

Untuk mengkonversi tanggal kalender masehi ke JD dapat dilakukan dengan menggunakan rumus berikut:

```
d = 367*Y - (7*(Y + ((M+9)/12)))/4 + (275*M)/9 + D - 730530
```

Rumus 2. 1 Konversi *Julian Date*

Dimana Y adalah tahun (dalam 4 digit), M bulan (1-12), D adalah tanggal.

2.2.3. Elemen Orbit

Elemen orbit utama pada tiap-tiap planet adalah :

```
N = longitude of the ascending node
i = inclination to the ecliptic (plane of the Earth's orbit)
w = argument of perihelion
a = semi-major axis, or mean distance from Sun
e = eccentricity (0=circle, 0-1=ellipse, 1=parabola)
M = mean anomaly (0 at perihelion; increases uniformly with time)
```

Elemen orbit dari Matahari :

```
N = 0.0
i = 0.0
w = 282.9404 + 4.70935E-5 * d
a = 1.000000  (AU)
e = 0.016709 - 1.151E-9 * d
M = 356.0470 + 0.9856002585 * d
```

Elemen orbit dari Bulan :

```
N = 125.1228 - 0.0529538083 * d
i = 5.1454
w = 318.0634 + 0.1643573223 * d
a = 60.2666  (Earth radii)
e = 0.054900
M = 115.3654 + 13.0649929509 * d
```

Elemen orbit dari Merkurius :

```
N =  48.3313 + 3.24587E-5 * d
i = 7.0047 + 5.00E-8 * d
w =  29.1241 + 1.01444E-5 * d
a = 0.387098  (AU)
e = 0.205635 + 5.59E-10 * d
M = 168.6562 + 4.0923344368 * d
```

Elemen orbit dari Venus :

```
N =  76.6799 + 2.46590E-5 * d
i = 3.3946 + 2.75E-8 * d
w =  54.8910 + 1.38374E-5 * d
a = 0.723330  (AU)
e = 0.006773 - 1.302E-9 * d
M =  48.0052 + 1.6021302244 * d
```

Elemen orbit dari Mars :

```
N =  49.5574 + 2.11081E-5 * d
i = 1.8497 - 1.78E-8 * d
w = 286.5016 + 2.92961E-5 * d
a = 1.523688  (AU)
e = 0.093405 + 2.516E-9 * d
M =  18.6021 + 0.5240207766 * d
```

Elemen orbit dari Yupiter :

```
N = 100.4542 + 2.76854E-5 * d
i = 1.3030 - 1.557E-7 * d
w = 273.8777 + 1.64505E-5 * d
a = 5.20256  (AU)
e = 0.048498 + 4.469E-9 * d
M =  19.8950 + 0.0830853001 * d
```

Elemen orbit dari Saturnus :

```
N = 113.6634 + 2.38980E-5 * d
i = 2.4886 - 1.081E-7 * d
w = 339.3939 + 2.97661E-5 * d
a = 9.55475  (AU)
e = 0.055546 - 9.499E-9 * d
M = 316.9670 + 0.0334442282 * d
```

Elemen orbit dari Uranus :

```
N =  74.0005 + 1.3978E-5 * d
i = 0.7733 + 1.9E-8 * d
w =  96.6612 + 3.0565E-5 * d
a = 19.18171 - 1.55E-8 * d  (AU)
e = 0.047318 + 7.45E-9 * d
M = 142.5905 + 0.011725806 * d
```

Elemen orbit dari Neptunus :

```
N = 131.7806 + 3.0173E-5 * d
i = 1.7700 - 2.55E-7 * d
w = 272.8461 - 6.027E-6 * d
a = 30.05826 + 3.313E-8 * d  (AU)
e = 0.008606 + 2.15E-9 * d
M = 260.2471 + 0.005995147 * d
```

2.2.4. Posisi Matahari

Untuk menghitung posisi matahari pertama-tama kalikan elemen orbit matahari dengan waktu JD.

```
w = 282.9404_deg + 4.70935E-5_deg   * d
a = 1.000000
e = 0.016709 - 1.151E-9             * d
M = 356.0470_deg + 0.9856002585_deg * d
```

Rumus 2. 2 Elemen Orbit Matahari

Lalu diperlukan *obliquity of the ecliptic* (sudut antara bidang orbit bumi dengan bidang ekuator bumi)

```
ecl = 23.4393_deg - 3.563E-7_deg * d
```

Rumus 2. 3 *Obliquity of Ecliptic*

Hitung *mean longitude* (L)

```
L = w + M
```

Rumus 2. 4 *Mean Longitude*

Hitung sudut *eccentric anomaly* (penyimpangan sudut)

```
E = M + (180/pi) * e * sin(M) * (1 + e * cos(M))
```

Rumus 2. 5 *Eccentric Anomaly*

Hitung koordinat matahari pada bidang orbit, dimana x menunjuk pada *perihelion*

```
xv = r * cos(v) = cos(E) – e
yv = r * sin(v) = sin(E) * sqrt(1 - e*e)
```

Rumus 2. 6 Koordinat Matahari Pada Bidang Orbit

Ubah manjadi jarak (r) dan *true anomaly* (v)

```
v = atan2( yv, xv )
r = sqrt( xv*xv + yv*yv )
```

Rumus 2. 7 Konversi Koordinat Menjadi Jarak dan *True Anomaly*

Hitung *longitude* dari matahari

```
lonsun = v + w
```

Rumus 2. 8 *Longitude* Matahari

Hitung koordinat *ecliptic* matahari

```
xs = r * cos(lonsun)
ys = r * sin(lonsun)
```

Rumus 2. 9 Koordinat *Ecliptic* Matahari

Putar ke koordinat ekuatorial dengan menggunakan ecl = 23.43721743_deg

```
xe = xs
ye = ys * cos(ecl)
ze = ys * sin(ecl)
```

Rumus 2. 10 Konversi Koordinat *Ecliptic* ke Koordinat Ekuatorial

2.2.5. Posisi Planet terhadap Matahari

Untuk menghitung posisi matahari pertama-tama kalikan setiap elemen orbit planet dengan waktu JD. Penghitungan dimulai dengan pendekatan pertama (E0 dan M dalam satuan derajat)

```
E0 = M + (180_deg/pi) * e * sin(M) * (1 + e * cos(M))
```

Rumus 2. 11 Pendekatan *Eccentric Anomaly*

Lalu lakukan pendekatan selanjutnya untuk memperkecil kesalahan

```
E1 = E0 - (E0 - (180_deg/pi) * e * sin(E0) - M) / (1 - e*cos(E0))
```

Rumus 2. 12 Pendekatan Lanjutan *Eccentric Anomaly*

Lakukan pendekatan hingga selisih nilai E0 dan E1 sama dengan 0.005. Dari hasil pendekatan tersebut dapat dihitung x,y koordinat orbit

```
x = r * cos(v) = a * (cos(E) - e)
y = r * sin(v) = a * sqrt(1 - e*e) * sin(E)
```

Rumus 2. 13 Perhitungan Koordinat Orbit

Lalu ubah menjadi jarak dan *true anomaly*

```
r = sqrt( x*x + y*y )
v = atan2( y, x )
```

Rumus 2. 14 Konversi Koordinat Menjadi Jarak dan *True Anomaly*

Setelah posisi planet pada jalur orbit sudah diketahui, langkah selanjutnya adalah mengubah posisi tersebut menjadi *eliptic coordinates*

```
x = r * (cos(N) * cos(v+w) - sin(N) * sin(v+w) * cos(i))
y = r * (sin(N) * cos(v+w) + cos(N) * sin(v+w) * cos(i))
z = r * sin(v+w) * sin(i)
```

Rumus 2. 15 Konversi Menjadi *Eliptic Coordinates*

2.3. Virtual Reality

Virtual Reality dipahami sebagai simulasi komputer interaktif yang dapat mempengaruhi indra pengguna bahkan menggantikan satu atau lebih indra manusia, sehingga pengguna larut kedalam lingkungan simulasi (*virtual environment*) [18].

Virtual Reality (VR) Secara bahasa berarti keadaan nyata/ide yang "dimasukkan" ke dalam dunia maya atau memvirtualkan objek nyata/ide yang tetap memperhitungakan sifat-sifat fisikanya. Oleh karena itu harus dibedakan dengan animasi 3D, yang terdapat pada film dan game, karena tidak memperhitungkan data dan kondisi fisik dari objek-objek yang berada di dalamnya (lingkungan virtual). Sebuah VR pasti memperhitungkan aspek ergonomis dan antropometri. Ini adalah added value sebuah VR. Ergonomis berarti barang yang divirtualkan harus cocok dengan anatomi tubuh manusia ketika digunakan seperti kita menggunakan barang-barang yang biasa berada di sekitar kita, sedangkan antopometri berarti di dalam virtualisasi tersebut diperhitungkan ukuran fisik dari gerakan manusia terhadap semua objek virtual di sekelilingnya. Dua hal tersebut merupakan aspek analisis yang menjadi pembeda VR terhadap games, aspek lainnya adalah fungsionalitas. Di sisi ini komponennya adalah *reachability*, *touchability*, dan *accessability*. *Reachability* berarti objek di dalam dunia vitual dapat dijangkau, dipegang, dapat berinteraksi dengan user. *Touchability* berarti objek dapat dirasakan, objek yang kita pegang atau sentuh memiliki berat ataupun kontur permukaan, dan *accessability* berarti objek dalam dunia virtual memiliki perilaku sama dengan objek dalam dunia nyata, misalkan bila dalam tubuh manusia terdapat 25 derajat kebebasan maka dalam dunia virtual pun harus sama. Gambar 2.4 menggambarkan penerapan teknologi VR yang digunakan oleh militer.



Gambar 2. 4 Pengguna Simulasi Virtual Reality

Virtual Reality merujuk pada pemakaian komputer untuk mensimulasikan sebuah pengalaman dengan cara yang sama dengan realita. Pada jenis-jenis VR yang paling sering dipakai, seseorang memakai sarung tangan, *earphone*, dan *goggles* yang disambungkan dengan komputer. Rangsangan berubah sesuai dengan gerakan orang itu, misalnya menggeleng-gelengkan kepala atau gerakan-gerakan lainnya.

Secara sederhana, *Virtual Reality* adalah pemunculan gambar-gambar tiga dimensi yang di bangkitkan komputer, yang terlihat nyata dengan bantuan sejumlah peralatan tertentu. Ciri terpentingnya adalah dengan menggunakan perangkat yang dirancang untuk tujuan tertentu, teknologi ini mampu menjadikan orang yang merasakan dunia maya tersebut terkecoh dan yakin bahwa yang dialaminya adalah nyata. Ada empat elemen dasar dari virtual reality [18], yaitu :

a. *Virtual Environment*

Definisi dari VR dan VE terus berkembang dan pada saat ini istilah keduanya saling berkaitan. VR mengarah kepada teknik atau sistem berupa perangkat dan software, sedangkan VE merupakan lingkungan yang diciptakan melalui komputer. *Virtual Environment* adalah lingkungan yang disimulasikan oleh

komputer, berupa lingkungan sebenarnya yang ditiru atau lingkungan yang hanya ada dalam imaginasi [18].

b. *Virtual Presence*

Virtual Presence, yaitu sebuah perasaan keberadaan seseorang dari lingkungan virtual. Pengguna tersebut bereaksi dengan objek virtual selayaknya berinteraksi dengan objek nyata. Pengguna merepresentasikan perasaan dari berada di sebuah lingkungan virtual. *Virtual presence* sulit untuk dimunculkan melalui media selain VR karena media lain tersebut tidak memberikan fasilitas atau kemampuan kepada sensor aktual dan imersi fisik langsung terhadap lingkungan. Pemahaman dari kehadiran langsung telah ditingkatkan melalui konsep dari kehadiran yang dimunculkan melalui media.

Imersi merupakan keadaan dimana pengguna berada di sebuah lingkungan yang berupaya untuk meningkatkan perasaan ruang atau perasaan seperti berada di keadaan nyata. *Virtual presence* dapat dikategorikan menjadi physical (sensory) dan mental presence [18].

1. ***Physical (sensory) Virtual Presence***

   Kehadiran virtual secara fisik mendefinisikan virtual reality dan sekaligus membedakannya dari media lain. Kehadiran virtual secara fisik ini didapat dari memberikan pengguna sebuah lingkungan virtual dengan satu atau lebih sensor yang dapat merubah posisi pengguna dan gerakannya. Pada umumnya sistem virtual reality melakukan render lingkungan virtual melalui penglihatan, pendengaran, dan sentuhan.

2. ***Mental Virtual Presence***

   Tingkatan kemampuan kehadiran virtual secara mental tergantung pada tujuan yang ingin dicapai melalui virtual reality. Jika virtual reality digunakan dengan tujuan hiburan, maka diperlukan kehadiran virtual secara mental dengan tingkat tinggi. Bagaimanapun, kehadiran virtual secara mental kadang tidak begitu dibutuhkan. Tidak adanya kehadiran virtual secara mental tidak mendiskualifikasi media dari menjadi virtual reality.

c. ***Sensory Feedback***

Umpan balik sensoris merupakan komponen krusial dari VR. Sistem VR memberikan umpan balik sensoris secara langsung melalui informasi visual. Sistem virtual reality memberikan umpan balik sensoris secara langsung kepada pengguna berdasarkan lokasi fisiknya.

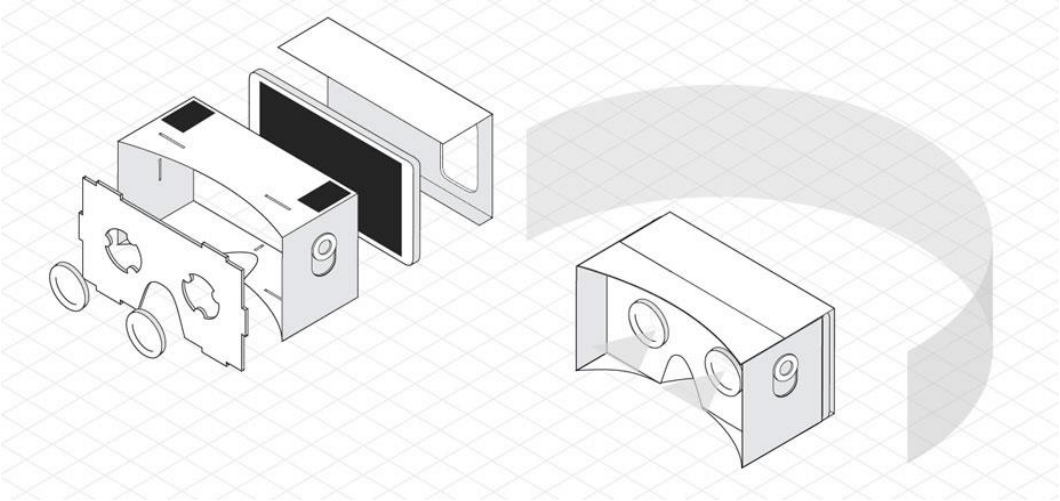
d. ***Interactivity***

Interaktivitas adalah salah satu fitur media baru yang paling banyak dibicarakan, mendapat tempat khusus di internet. Seperti halnya berbagai istilah dalam dunia cyber baru, kadang sulit memahami arti sebenarnya dari kata itu. Satu masalah dalam mendefinisikan istilah interaktivitas adalah bahwa ia dipakai minimal dalam dua makna berbeda. Orang-orang dengan latar belakang ilmu komputer cenderung memaknainya sebagai interaksi pengguna dengan komputer, sebagaimana permainan-permainan interaktif. Definisi semacam itu menyebutkan bahwa interaktivitas berarti kemampuan pengguna untuk berkomunikasi secara langsung dengan komputer dan memiliki dampak pada pesan apapun yang sedang dibuat. Para sarjana komunikasi cenderung berpikir bahwa interaktivitas merupakan komunikasi antara dua manusia. William, Rice, dan Rogers (1988) mendefinisikan interaktivitas sebagai tingkatan dimana pada proses komunikasi para partisipan memiliki kontrol terhadap peran dan dapat bertukar peran dalam dialog mutual mereka. Interaktivitas memiliki 6 dimensi, yaitu:

1. Persuasi – menginformasikan
2. Kontrol lemah – kontrol tinggi
3. Aktivitas rendah – aktivitas tinggi
4. Satu arah – dua arah
5. Waktu tertentu – waktu fleksibel
6. Kesadaran rendah terhadap tempat – kesadaran tinggi terhadap tempat

2.4. *Google Cardboard*

Google Cardboard adalah wahana virtual reality yang dikembangkan oleh Google dengan bahan karton yang dilipat dan menggunakan *handphone* sebagai layarnya, lihat gambar 2.5. Contoh penggunaannya adalah kita jika kita bermain game VR, maka seolah-olah kita berada di dunia game tersebut, kemudian menonton video 3D, menonton film seakan-akan kita berada di bioskop, *virtual tour*, menonton film 360° (seakan-akan kita berada disana).



Gambar 2. 5 *Google Cardboard* [19]

2.5. Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware*, dan aplikasi. Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan *platform* terbuka bagi para pengembang untuk membuat aplikasi mereka sendiri.

Pada awalnya dikembangkan oleh Android Inc, sebuah perusahaan pendatang baru yang membuat perangkat lunak untuk ponsel yang kemudian dibeli oleh Google Inc. Untuk pengembangannya, dibentuklah *Open Handset Alliance* (OHA), konsorsium dari 34 perusahaan perangkat keras, perangkat lunak, dan telekomunikasi termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

2.5.1. Versi Android

Sistem operasi android memiliki beberapa versi, diantaranya adalah sebagai berikut [20] :

a. Android versi 1.5 (*Cupcake*)

Android versi 1.5 dirilis pada pertengahan Mei 2009, memiliki beberapa penambahan fitur dalam seluler yakni kemampuan merekam dan menonton *video* dengan modus kamera, mengunggah *video* ke *youtube* dan gambar ke *picasa* langsung melalui telepon.

b. Android versi 1.6 (*Donut*)

Pada September 2009, Google merilis android versi 1.6 dengan kemampuan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol *applet* VPN.

c. Android versi 2.0 (*Éclair*)

Android versi 2.0 dirilis pada 3 Desember 2009, perubahan yang dilakukan adalah dengan mengoptimalkan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML 5, daftar kontak baru, dukungan *flash* untuk kamera 3,2 MP, *digital zoom*, dan *Bluetooth* 2.1.

d. Android versi 2.2 (*Froyo*)

Pada 20 Mei 2010, Google merilis android versi 2.2 dengan penambahan dukungan adobe flash 10.1, integrasi V8 *javascript engine* yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam SD Card, kemampuan Wifi Hotspot portable, dan kemampuan auto update dalam aplikasi android market.

e. Android versi 2.3 (*Gingerbread*)

Android versi 2.3 diluncurkan pada tanggal 6 Desember 2010 dengan peningkatan kemampuan permainan (gaming), peningkatan fungsi copy paste, perubahan UI (User Interface), dukungan kemampuan NFC (Near Field Communication), dan dukungan jumlah kamera lebih dari satu.

f. Android versi 3.0 (*Honeycomb*)

Android versi ini dirancang khusus untuk perangkat tablet dengan dukungan ukuran layar yang lebih besar. Honeycomb juga mendukung multi proses dan akselerasi perangkat keras untuk grafis.

g. Android versi 4.0 (*Ice Cream Sandwich*)

Pada tanggal 19 Oktober 2011, android versi 4.0 dirilis dengan membawa fitur dari versi 3.0 dan menambahkan fitur baru diantaranya membuka kunci dengan pengenalan wajah, pengontrol penggunaan jaringan data, sinkronisasi kontak, perangkat tambahan fotografi, dan berbagi informasi dengan menggunakan NFC.

h. Andoid versi 4.1 (*Jelly Bean*)

Android versi 4.1 dirilis dengan penambahan fitur baru, diantaranya dukungan terhadap OpenGL ES 3.0 yang menjadikan performansi tinggi pada sektor grafis, selain itu terdapat juga fitur bluetooth smart yang dapat menghemat daya pada saat pemakaian bluetooth.

i. Andoid versi 4.4 (*KitKat*)

Android versi 4.4 dirilis pada tanggal 31 Oktober 2013 dengan pembaharuan antarmuka, optimasi kinerja pada perangkat dengan spesifikasi rendah, peningkatan tampilan mode layar penuh, dan dukunagn Bluetooth Message Access Profile (MAP).

j. Android versi 5.0 (*Lollipop*)

Android versi 5.0 dirilis pada tanggal 3 November 2014 dengan perbaharuan antarmuka dengan warna yang lebih hidup. Terdapat beberapa fitur baru, diantaranya penghemat baterai, device sharing, notifikasi, desain materia, dan keamanan yang lebih baik.

2.6. *Accelerometer*

Accelerometer berfungsi untuk mengukur percepatan, mendeteksi getaran, dan bisa juga untuk percepatan gravitasi. Pendeteksian gerakan berdasarkan pada 3 sumbu yaitu kanan-kiri, atas-bawah dan depan-belakang. Pengaplikasian sensor ini biasanya untuk pengukuran kecepatan mesin, getaran mesin, getaran pada bangunan dan kecepatan yang disertasi dengan pengaruh gravitasi bumi. Gambar 2.6 menunjukan bentuk dari *accelerometer.* [21]



Gambar 2. 6 *Accelerometer*

Prinsip kerja accelerometer berdasarkan pada medan magnet yang digerakkan pada suatu konduktor ataupun konduktor yang digerakkan pada medan magnet maka akan timbul induksi elektromagnetik pada konduktor tersebut. Gambar 2.7 menunjukan prinsip kerja dari *accelerometer* pada *handphone.*



Gambar 2. 7 Prinsip Kerja *Accelerometer* pada *Handphone*

2.7. *Gyroscope*

Gyroscope berfungsi untuk mengukur/menentukan orientiasi suatu benda berdasarkan pada ketetapan momentum sudut. Dari pengertian lain *gyroscope* berfungsi untuk menentukan gerakan sesuai dengan gravitasi yang dilakukan oleh pengguna. *Gyroscope* ini memiliki peranan yang sangat penting dalam hal mempertahankan keseimbangan suatu benda seperti penggunaannya pada pesawat terbang yang dapat menentukan kemiringan pada sumbu x,y, dan z.

Output yang dihasilkan oleh *gyroscope* berupa kecepatan sudut yang pada sumbu x akan menjadi phi (Φ), sumbu y menjadi theta (θ), dan sumbu z menjadi psi (Ψ). Sebelum digunakan biasanya *gyroscope* di kalibarasi terlebih dahulu dengan menggunakan bandul yang fungsinya untuk menentukan nilai faktor ataupun dapat juga melihat pada datasheet sensor yang digunakan.

Prinsip kerja dari *gyroscope* ini adalah pada saat *gyroscope* berotasi maka *gyroscope* akan memiliki nilai keluaran. Apabila *gyroscope* berotasi searah dengan jarum jam pada sumbu Z maka tegangan *ouput* yang dihasilkan akan mengecil sedangkan jika *gyroscope* berotasi berlawan arah dengan jarum jam pada sumbu Z maka tegangan *output* yang dihasilkan akan membesar. Pada saat *gyroscope* tidak sedang berotasi atau berada pada keaadaan diam maka tegangan *ouputnya* akan sesuai dengan nilai *offset gyrosensor* tersebut. Gambar 2.8 menunjukan prnsip kerja dari *gyroscope* dimana sisi terluar mewakili sumbu y, sisi tengah mewakili sumbu z, dan sisi terdalam mewakili sumbu x. [21]



Gambar 2. 8 Prinsip Kerja *Gyroscope*

2.8. Unity 3D

Unity adalah sebuah ekosistem dari pengembangan *game*, mesin *render* yang terintegrasi penuh dengan satu set peralatan yang intuitif dan alur kerja yang cepat untuk menciptakan konten 3D dan 2D yang interaktif, mudah dipublikasikan ke berbagai *platform*, aset tersedia di *asset store*, dan memiliki komunitas untuk berbagi pengetahuan [22].

Adapun fitur-fitur yang dimilik oleh Unity 3D antara lain sebagai berikut:

a. *Integrated development environment* (IDE) atau lingkungan pengembangan terpadu.
b. Penyebaran hasil aplikasi pada banyak *platform*.
c. *Engine* grafis menggunakan Direct3D (Windows), OpenGL (Mac, Windows), OpenGL ES (iOS), and *proprietary* API (Wii).

d. *Game Scripting* melalui Mono. *Scripting* yang dibangun pada Mono, implementasi *open source* dari NET Framework. Selain itu Pemrogram dapat menggunakan UnityScript (bahasa kustom dengan sintaks JavaScriptinspired), bahasa C # atau Boo (yang memiliki sintaks Python-inspired).

Mesh merupakan bentuk dasar dari obyek 3D. Pembuatan *mesh* tidak dilakukan pada Unity. Sementara *GameObjects* adalah kontainer untuk semua Komponen lainnya. Semua objek dalam permainan disebut *game objects*.

Material digunakan dan dihubungkan dengan *mesh* atau *renderer* partikel yang melekat pada *game object*. Material berhubungan dengan penyaji *Mesh* atau partikel yang melekat pada *GameObject* tersebut. Mereka memainkan bagian penting dalam mendefinisikan bagaimana objek ditampilkan. *Mesh* atau partikel Tidak dapat ditampilkan Tanpa material karena material meliputi referensi untuk *Shader* yang digunakan untuk membuat *Mesh* atau Partikel. *Material* digunakan untuk menempatkan Tekstur ke *GameObjects*.

Unity mendukung pengembangan aplikasi Android. Sebelum dapat menjalankan aplikasi yang dibuat dengan Unity Android diperlukan adanya pengaturan lingkungan pengembang Android pada perangkat. Untuk itu pengembang perlu men-download dan menginstal SDK Android dan menambahkan perangkat fisik ke sistem. Unity Android memungkinkan pemanggilan fungsi kustom yang ditulis dalam C / C + + secara langsung dan Java secara tidak langsung dari script C #.

Perbedaan mendasar antara unity desktop dan unity android adalah sebagai berikut:

1. *Dynamic typing* pada *javascript* tidak diperbolehkan dalam unity android
2. *Terrain Engine* tidak didukung pada perangkat android
3. ETC sebagai *texture compression* dikarenakan android tidak mendukung tekstur DXT/ PVRTC/ ATC.
4. *Movie texture* tidak didukung pada android, tetapi *streaming video* layar penuh disediakan melalui fungsi *scripting*.

2.9. Autodesk 3ds Max

3ds Max merupakan sebuah *software modeling* 3D yang menyediakan model komprehensif, animasi, simulasi, dan *rendering* solusi untuk *game*, film, dan gerak seniman grafis. 3ds Max memberikan alat-alat baru yang efisien, mempercepat kinerja, dan alur kerja yang efisien untuk membantu meningkatkan produktivitas secara keseluruhan untuk bekerja dengan kompleks, aset-resolusi tinggi.

2.10. Adobe Photoshop

Adobe Photoshop merupakan program pengolah grafik yang mampu bekerja pada dua tipe grafik yaitu bitmap dan vektor. Oleh sebab itu, file kerja pada Adobe Photoshop dapat berupa gambar bitmap maupun vektor. Hal ini merupakan keunggulan dari program Adobe Photoshop karena dengan kemampuan tersebut akan memudahkan untuk membuat obyek, mengolah foto maupun pengeditan foto lebih lanjut.

2.11. Skala

Terdapat dua teknik membuat skala yang sering digunakan dalam riset dengan data kualitatif yaitu [23]:

2.11.1. Skala Likert

Skala likert berhubungan dengan pernyataan tentang sikap seseorang terhadap sesuatu, misalnya setuju-tidak setuju, senang-tidak senang, dan baik-tidak baik. Responden diminta mengisi pernyataan dalam sakala ordinal berbentuk verbal dalam jumlah kategori tertentu bisa 5,7 (agar dapat menampung kategori yang “netral”) atau memasukan kategori “tidak tahu”.

2.11.2. Skala Guttman

Skala Guttman hanya mengukur satu dimensi dari suatu variabel yang memiliki beberapa dimensi, selain itu skala ini pun merupakan bentuk skala kumulatif.

2.12. *Quota Sampling*

Teknik *quota sampling* adalah teknik pengambilan sampel dengan cara menetapkan jumlah tertentu sebagai target yang harus dipenuhi dalam pengambilan sampel dari populasi (khususnya yang tidak terhingga atau tidak jelas), kemudian dengan patokan jumlah tersebut peneliti mengambil sampel secara sembarang asal memenuhi persyaratan sebagai sampel dari populasi tersebut. [24]

Pada *quota sampling* banyaknya sampel yang ditetapkan itu hanya sekedar perkiraan akan relatif memadai untuk mendapatkan data yang diperlukan yang diperkirakan dapat mencerminkan populasinya, tidak bisa diperhitungkan secara tegas proporsinya dari populasi, karena jumlah anggota populasi tidak diketahui secara pasti tadi.

2.13. Uji Kuesioner

Sebelum data dianalisis, terlebih dahulu dilakukan pengolahan data. Setelah data terkumpul melalui kuesioner maka langkah selajutnya adalah melakukan tabulasi, yaitu memberikan nialai (*Scorring*) sesuai dengan sistem yang diterapkan. *Scorring* dilakukan dengan menggunakan skala Likert 5-4-3-2-1.

2.13.1. Uji Validitas

Uji validitas adalah pengujian suatu alat untuk menentukan kecermatan alat ukur tersebut dalam melakukan fungsi ukurnya. Maksudnya pengujian ini adalah untuk menguji apakah tiap pertanyaan kuesioner benar-benar mengungkapkan factor-faktor yang ingin dianalisis. Uji validitas berguna untuk mengetahui apakah ada pertanyaan-pertanyataan pada kuesioner yang harus diganti atau dibuang karena dianggap tidak relevan [25].

Kriteria yang digunakan valid atau tidak valid adalah jika nilai pada total korelasi antara skor masing-masing butir pertanyaan mempunyai tingkat signifikasi diatas 0,30 maka butir pertanyaan tersebut dikatakan valid, dan jika korelasi skor masing-masing butir pertanyaan tersebut dikatakan tidak valid. pengujian validitas tiap butir menggunakan analisis item, yaitu mengkorelasikan skor tiap butir dengan skor total yang merupakan jumlah tiap skor butir.

2.13.2. Uji Reabilitas

Reliabilitas menunjuk pada suatu pengertian bahwa sesuatu instrumen cukup dapat dipercaya untuk diinginkan sebagai alat pengumpul data karena instrumen tersebut sudah baik. Instrumen yang tidak baik akan bersifat tendensius mengarahkan responden untuk memilih jawaban-jawaban tertentu. Instrumen yang sudah dapat dipercaya, yang realibel akan menghasilkan data yang dapat dipercaya juga. Uji Reliabilitas adalah alat ukur untuk mengukur suatu kuesioner yang merupakan indikator dari variabel atau konstruk. Uji reliabilitas ini digunakan untuk menguji konsistensi data dalam jangka waktu tertentu, yaitu untuk mengetahui sejauh mana pengukuran yang digunakan dapat dipercaya atau diandalkan. Uji realibilitas ini bertujuan untuk melihat konsistensi alat ukur [26].

Teknik yang digunakan untuk uji realibilitas adalah *Cronbach Alpha*. Teknik ini dikembangkan oleh Cronbach untuk menghasilkan korelasi realibilitas alpha, dan merupakan teknik pengujian konsistensi realibilitas antara item-item terpopuler, serta menunjukan indeks konsistensi yang sempurna. Suatu konstruk atau variabel dikatakan *reliable* jika memberikan nilai *Cronbach Alpha* > 0,06 .

$$\text{Koefisien } Cronbach\ Alpha\ (r) = \left[\frac{k}{(k-1)}\right]\left[1 - \sum \frac{\sigma_b^2}{\sigma_i^2}\right]$$

Rumus 2. 16 Koefisien *Cronbach Aplha*

Keterangan :

r : Koefesien *Cronbach Alpha*
k : Banyaknya pertanyaan
$\sum$: Jumlah semua item yang diuji
σ_b^2 : *Variance of responses to a lestitem*
σ_i^2 : *Variance of total test score*

Berdasarkan jawaban responden terhadap kuesioner dilakukan analisis deskriptif. Analisis deskriptif didasarkan pada penilaian yang dicari dengan cara rentang kretria [25].

a. Analisis Tabulasi Sederhana

Data yang diperoleh diolah kedalam bentuk persentase dengan mengunakan rumus :

$$P = \frac{fi}{\Sigma fi} \text{ x } 100\%$$

Rumus 2. 17 Analisis Tabulasi Sederhana

Dimana :

P = Persentase responden yang memilih katagori tertentu.

fi = jumlah responden yang memilih katagori tertentu.

Σfi = banyak jumlah responden

b. Skor Rata - Rata

Setiap jawaban responden dari pertanyan yang diberikan bobot. Cara menghitung skor adalah menjumlahkan seluruh hasil kali masing-masing bobot dibagi dengan jumlah total seluruh frekuensi.

$$x = \frac{\Sigma fiWi}{\Sigma fi}$$

Rumus 2. 18 Perhitungan Rata-rata Bobot

Dimana:
x = rata-rata berbobot
fi = frekuensi
wi = bobot

Tabel 2. 3 Skor Penilaian

Kriteria Penilaian	**Skor**
Sangat Tidak Setuju (STS)	= 1
Tidak Setuju (TS)	= 2
Ragu-ragu (RR)	= 3
Setuju (S)	= 4
Sangat Setuju (SS)	= 5

Setelah itu gunakan rentang skala maka skala penilaian untuk menentukan posisi tanggapan responden dengan menggunakan nilai skor setiap variabel. Bobot

alternatif jawaban yang terbentuk dari teknik skala peringkat terdiri dari kisaran 1 hingga 5.

Perhitungan rentang skala dengan menggunakan rumus dapat dilihat pada rumus 2.19 [27]:

$$\text{Rs} = \frac{R(bobot)}{M}$$

Rumus 2. 19 Perhitungan Rentang Skala

Dimana :

Rs = rentang skala

R(bobot) = Bobot tertinggi – Bobot Terendah

M = Banyaknya Kategori Bobot

Rentang skala *likert* yang digunakan dalam penelitian ini adalah 1 hingga 5, maka rentang skala penilaian yang didapat adalah :

$$Rs = \frac{5-1}{5} = 0{,}8$$

Tabel 2. 4 Skala Penilaian

Penilaian	**Skala**
Sangat Tidak Setuju	= 1,00 – 1,80
Tidak Setuju	= 1,81 – 2,60
Ragu	= 2,61 – 3,40
Setuju	= 3,41 – 4,20
Sangat Setuju	= 4,21 – 5,00

2.14. *Black-Box Testing*

Pengujian *black-box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black-box* merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalahan daripada metode *white-box*. [28]

Pengujian *black-box* berusaha menemukan kesalahan dalam kategori sebagai berikut :

1. Fungsi – fungsi yang tidak benar atau hilang.
2. Kesalahan Interface.
3. Kesalahan dalam struktur data atau akses database eksternal.
4. Kesalahan Kinerja.
5. Inisialisasi dan kesalahan terminasi.

Tidak seperti pengujian *white-box*, yang dilakukan pada saat awal proses pengujian, pengujian *black-box* cenderung diaplikasikan selama tahap akhir pengujian. Karena pengujian *black-box* memperhatikan struktur kontrol, maka perhatian berfokus pada domain informasi. Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut :

1. Bagaimana validasi fungsional diuji?
2. Kelas input apa yang akan membuat test case menjadi baik?
3. Apakah sistem sangat sensitive terhadap harga input tertentu?
4. Bagaimana batasan dari suatu data diisolasi?
5. Kecepatan data apa dan volume data apa yang akan dapat ditolerir oleh sistem?
6. Apa pengaruh kombinasi tertentu dari data terhadap operasi sistem?

Dengan mengaplikasikan teknik black-box, maka kita menarik serangkaian test case yang memenuhi criteria berikut ini :

1. *Test case* yang mengurangi, dengan harga lebih dari satu, jumlah *test case* tambahan yang harus didesain untuk mencapai pengujian yang dapat dipertanggung jawabkan,
2. *Test case* yang member tahu kesalahan yang berhubungan hanya dengan pengujian spesifik yang ada.

2.15. UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan

mendokumentasikan sistem piranti lunak [29]. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa pemrograman berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk *modeling* aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan.

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma

menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek [29].

Melalui berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML, sebenarnya konsepsi dasar UML dapat dirangkum seperti pada tabel 2.5.

Tabel 2. 5 Konsep Dasar UML

Major Area	***View***	***Diagrams***	***Main Concepts***
Structural	*Static view*	*Class diagram*	*Class,association,generalization, dependency, realization, interface*
	Use case view	*Use case diagram*	*Use case, actor, assosiation, extend, include, use case generalization*
	Implementation view	*Component diagram*	*Component, interface, dependency, realization*
	Deployment view	*Deployment diagram*	*Node, component, dependency, location*
Dynamic	*State machine view*	*Statechart diagram*	*State, event, transition, action*
	Activity view	*Activity diagram*	*State, activity, completion transition, fork, join*
	Interaction view	*Sequence diagram*	*Interaction, object, message, activation*
		Collaboration diagram	*Collaboration, interaction, collaboration role, message*
Model management	*Model management view*	*Class diagram*	*Package, subsystem, model*
Extensibility	*All*	*All*	*Constraint, stereotype, tagged values*

Abstraksi konsep dasar UML yang terdiri dari *structural classification*, *dynamic behavior*, dan *model management*, dapat dipahami dengan mudah apabila melihat tabel 2.2. *Main concepts* dapat dipandang sebagai *term* yang akan muncul pada saat membuat diagram, sedangkan *view* adalah kategori dari diagram tersebut.

2.15.1. Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, membuat sebuah daftar belanja, dan sebagainya. Seorang/sebuah actor adalah sebuah *entitas* manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu [29].

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat menambahkan fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang ditambahkan akan dipanggil setiap kali *use case* yang menambahkan dieksekusi secara normal.

Sebuah *use case* dapat ditambahkan oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat melakukan *extend use case* lain dengan behaviour yang dimiliki. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

2.15.2. *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi) [29].

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

a. Nama (dan *stereotype*)
b. Atribut

c. Metode

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak- anak yang mewarisinya.
c. *Public*, dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*. Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.

Hubungan Antar *Class* :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas”). Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
3. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence* diagram yang akan dijelaskan kemudian.

2.15.3. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga

dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi [29].

Activity diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari *level* atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity* diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

2.15.4. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait) [29].

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang menjadi *trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal.

Message digambarkan sebagai garis berpanah dari satu objek ke objek

lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity*.

2.16. *Voice Recognition*

Voice Recognition adalah suatu sistem yang dapat mengidentifikasi seseorang melalui suaranya, *voice recognition* menjadi dua jenis yaitu *speech recognition* dan *speaker recognition*.

2.16.1. *Speech Recognition*

Speech Recognition adalah proses identifikasi suara berdasarkan kata yang diucapkan dengan melakukan konversi sebuah sinyal akustik, yang ditangkap oleh *audio device* (perangkat input suara). *Speech recognition* juga merupakan sistem yang digunakan untuk mengenali suara perintah kata dari suara manusia dan kemudian diterjemahkan menjadi suatu data yang dimengerti oleh komputer. Pada saat ini, sistem ini digunakan untuk menggantikan peranan input dari keyboard dan mose.

Keuntungan dari sistem ini adalah pada kecepatan dan kemudahan dalam penggunaan nya. Kata-kata yang ditangkap dan dikenali bisa jadi sebagai hasil akhir, untuk sebuah aplikasi seperti *command* dan *control*, pengimputan data, dan persiapan dokumen. Parameter yang dibandingkan ialah tingkat penekenan suara yang kemudian akan dicocokan dengan template database yang tersedia [30].

2.16.2. *Speaker Recognition*

Speaker recognition adalah suatu proses pengenalan pembicara dari informasi yang terkandung dalam gelombang suara yang diinputkan. *Speaker recognition* dibagi menjadi dua bagian, yaitu *speaker verification* dan *speaker identification*. *Speaker verification* adalah proses pemeriksaan ulang seorang pembicara, sehingga perlu diketahui terlebih dahulu identitas pembicara tersebut berdasarkan data yang telah dimasukkan (misalnya *username* dan *password*). *Speaker verification* membandingkan fitur suara dari seorang pembicara secara

langsung dengan fitur pembicara tertentu yang ada didalam database. Bila hasil perbandingan tersebut lebih besar atau sama dengan batasan tertentu (*threshold*), maka pembicara tersebut diterima, bila tidak maka ditolak. *Speaker identification* adalah proses untuk mencari dan mendapatkan identitas dari seorang pembicara dengan membandingkan fitur suara yang dimasukkan dengan semua fitur suara pembicara yang ada didalam database. Berbeda dengan pada *speaker verification*, proses ini melakukan perbandingan *one-to-many* (1:N). Bila dibandingkan dengan *speaker verification*, maka *speaker identification* lebih sulit. Hal ini disebabkan, bila jumlah pembicara yang terdaftar dalam database semakin besar, maka kemungkinan terjadinya kesalahan dalam pengambilan keputusan semakin besar pula. Lebih jauh *speaker recognition* dapat diklasifikasikan menjadi *text-dependent* dan *text independent.* Pada *text-dependent*, kata-kata yang diucapkan oleh pembicara sudah ditentukan sebelumnya. Sedangkan pada *text-independent*, tidak ada asumsi kata-kata yang akan diucapkan oleh pembicara, sehingga sistem harus memodelkan fitur-fitur umum dari suara seorang pembicara [31].

Pada *speaker recognition* ada beberapa faktor yang dapat menyebabkan kesalahan dalam proses verifikasi dan identifikasi, antara lain :

1. Kesalahan dalam pengucapan (*misspoken*) dan pembacaan (*missread*) frasa.
2. Keadaan emosional yang ekstrim (misalnya stres).
3. Pergantian penempatan mikrofon (*intrasession* atau *intersession*).
4. Kekurangan atau ketidak-konsistenan akustik dari ruangan (misalnya multipath dan noise).
5. *Channel mismatch* (misalnya penggunaan mikrophon yang berbeda *channel* dalam perekaman dan verifikasi).
6. Sakit (misalnya flu yang dapat merubah *vocal tract*).
7. *Aging* (model *vocal tract* dapat berubah berdasarkan usia).

2.16.3. *Fast Fourier Transform*

Setiap frame dikonversi dari domain waktu ke domain frekuensi untuk mendapatkan spektrum frekuensinya. Hal ini dilakukan untuk mempermudah komputasi dan analisa. Metode yang dilakukan untuk mendapatkan spektrum frekuensi adalah *Fast Fourier Transform (FFT)* [32].

Berikut adalah rumusan untuk perhitungan FFT :

$$X_n = \sum_{k=0}^{N-1} x_k e^{-2\pi jkn/N} \text{ , n=0,1,2,....,N-1}$$

Rumus 2. 20 Pencarian *Fast Fourier Transform*

Keterangan :

x_k : Deretan aperiodik dengan nilai N

N : Jumlah sampel (kelipatan 2)