

BAB 2

LANDASAN TEORI

2.1 Pengertian Sistem

Sistem adalah kumpulan beberapa elemen yang saling berinteraksi dan berkomunikasi untuk mencapai suatu tujuan tertentu. Sebuah sistem akan memberikan gambaran tentang suatu kejadian nyata di lapangan berupa objek seperti tempat, benda dan pengguna yang terlibat didalamnya. Jadi, sistem merupakan gabungan dari bagian-bagian yang memiliki keterkaitan, saling bekerjasama membentuk satu kesatuan untuk mencapai target yang diinginkan[8].

2.2 Karakteristik Sistem

Sistem merupakan kumpulan unit-unit dalam satu tujuan, karenanya sistem memiliki sifat atau karakteristik sebagai berikut :

1. Masukan (*Input*)

Masukan adalah energi atau data yang dimasukkan kedalam sistem.

2. Komponen (*Component*)

Setiap sistem terdiri dari komponen pendukung yang saling bekerjasama dan berinteraksi membentuk satu kesatuan.

3. Batasan Sistem (*Boundary*)

Batasan merupakan penjelas untuk memisahkan interaksi suatu sistem dengan sistem yang lain.

4. Lingkungan Sistem (*Environment*)

Lingkungan sistem merupakan batasan dari sistem berupa hal-hal yang dapat mempengaruhi operasi yang terjadi didalam sistem.

5. Pengolahan (*Process*)

Sebuah sistem akan dapat mempelajari suatu bagian pemrosesan yang akan mengubah masukan menjadi keluaran.

6. Penghubung Sistem (*Interface*)

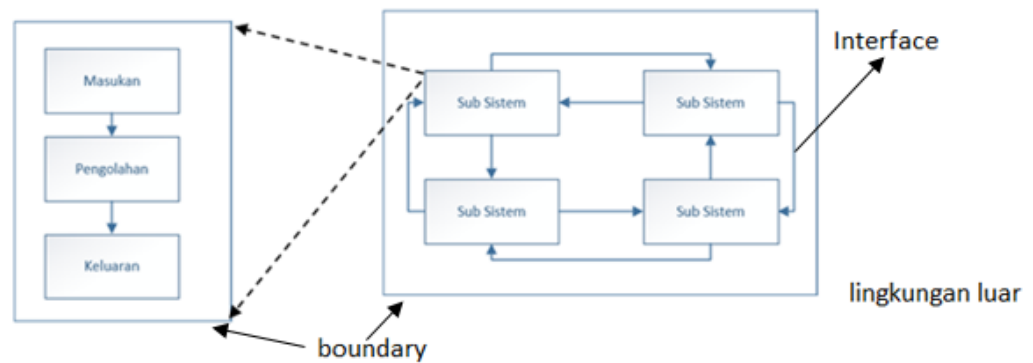
Sistem yang terdiri dari bagian sub sistem memiliki media penghubung antar komponen untuk melakukan akses.

7. Keluaran (*Output*)

Hasil dari pengolahan data masukan akan menjadi keluaran berupa informasi yang memiliki nilai.

8. Tujuan Sistem (*Goal*)

Tujuan sistem akan sangat menentukan segala aktivitas yang terjadi dalam sistem seperti kebutuhan masukan dan nilai dari hasil keluaran.[9]



Gambar 2. 1 Karakteristik Sistem

2.3 Pengertian Monitoring

Monitoring merupakan prosedur secara deskriptif yang digunakan untuk mengukur pengaruh dan mengidentifikasi aktivitas yang sedang terjadi. Monitoring juga dapat diartikan sebagai pengawasan sebuah proses yang terjadi untuk kemudian menentukan hal apa yang harus dikerjakan, agar tercapainya suatu tujuan yang diinginkan. Data akhir dari proses pemantauan harus diolah terlebih dahulu agar dapat dimaknai dengan segera, yaitu supaya diketahui apakah tujuan pelaksanaan program sesuai dengan yang direncanakan.

Adapun tujuan dari aktivitas monitoring adalah sebagai berikut:

1. Mengidentifikasi masalah yang timbul agar segera dapat diatasi.
2. Untuk mengetahui keterkaitan antara kegiatan yang sedang terjadi dengan tujuan yang direncanakan agar dapat memperoleh ukuran kemajuan.
3. Mempelajari kegiatan yang dilaksanakan apakah sesuai dengan rencana akhir.
4. Melakukan identifikasi pola kerja yang dilakukan apakah sudah tepat agar dengan mudah mencapai tujuan.

5. Menyesuaikan kegiatan dengan lingkungan yang berubah, tanpa menyimpang dari tujuan sebenarnya [10]

2.4 Tanaman Stroberi

Stroberi termasuk tanaman dua tahunan, buah ini hanya diproduksi di daerah tertentu, umumnya di dataran tinggi [3]. Stroberi yg kita temukan di pasar swalayan adalah hibrida yg dihasilkan dari persilangan *F. virginiana* L. var Duchesne asal Amerika Utara dgn *F. chiloensis* L. var Duchesne asal Chili. Persilangan itu menghasilkan hibrid yg merupakan stroberi modern (komersil) *Fragaria x annanassa* var Duchesne.

Tabel 2. 1 Klasifikasi Botani Tanaman Stroberi

Divisi	Spermatophyta
Sub divisi	Angiospermae
Kelas	Dicotyledonae
Keluarga	Rosaceae
Genus	<i>Fragaria</i>
Spesies	<i>Fragaria</i> spp.

2.4.1 Ciri-ciri Tanaman Stroberi

1. Batang Utama

Batang tanaman strawberry sangat pendek. Daun-daun terbentuk disetiap buku. Pada ketiak daun terdapat pucuk aksilar. Internode sangat pendek sehingga jarak daun yang satu dengan yang lain sangat rapat. Tanaman seperti rumput tanpa batang. Batang utama dan daun yang tersusun rapat tersebut disebut crown, ukuran crown berbeda-beda, tergantung dari umur, tingkat perkembangan tanaman, kultivar dan kondisi lingkungan pertumbuhan [3].

2. Daun

Dalam masa pertumbuhan vegetatif, meristem apikal membentuk daun-daun baru setiap 8-12 hari pada suhu rata-rata 22°C. Daunnya dapat bertahan selama 1-3 bulan, kemudian kering. Pada daun stroberi terdapat

stomata yang jumlahnya sekitar 300-400 stomata. Hal ini mengakibatkan daun stroberi banyak kehilangan air melalui transpirasi [3].

3. Akar

Tanaman stroberi dewasa umumnya mempunyai 20-35 akar primer dengan panjang akar sekitar 40 cm. Namun, ada juga jenis stroberi yang mempunyai 100 akar perimer. Akar perimer dapat bertahan lebih dari satu tahun. Akar-akar baru yang menggantikan akar primer tumbuh dari ruas yang apaling dekat dengan akar perimer. Hal ini dapat mengurangi kontak akar dengan tanah pada tanamantanaman tua. Akar-akarnya berkumpul dengan panjang 0.5 m. Sekitar 90% dari total akar berkumpul di kedalaman antara 15-45 cm [3].

4. Bunga

Bunga tanaman stroberi mempunyai 5 *sepal* (kelopak bunga), 5 petal (daun mahkota), 20-35 *stemen* (benang sari), dan ratusan *pistil* (putik) yang menempel pada *receptacle* (dasar bunga) dengan pola melingkar. Bunga tersusun dalam *infloresens* (malai) yang terletak di ujung tanaman.pada kondisi pertumbuhan yang cocok, crown cabang yang muncul dari ketiak daun terakhir akan mebentuk bunga pada ujungnya sehingga timbul kesan dua infloresens dalam satu tanaman [3].

5. Buah

Buah stroberi berwarna merah. Buah yang biasa dikenal adalah buah semu yang sebenarnya merupakan *receptacle* yang membesar. Buah sejati yang berasal dari *ovul* yang telah diserbuki berkembang menjadi buah kering dengan biji keras. Struktur buah keras ini disebut *achene*. Buah sejati ini berukuran kecil dan menempel pada *receptacle* yang membesar. Ukuran stroberi ditentukan oleh jumlah pistil dan keefektifan penyerbukan [3].

2.4.2 Gejala Penyakit Tanaman Stroberi

1. Hawar Daun

Gejala hawar dimulai dari tepi daun menuju ke tengah dan daun akan menjadi hijau kusam. gejala semakin berat ditunjukkan dengan daun yang menjadi layu dan mengering. Serangan yang parah menyebabkan seluruh

tanaman tumbuh meranggas dan mati. Pada cuaca panas, tanaman akan menjadi cepat layu dan mati. Gejala ini banyak terlihat pada bibit yang baru dipindah semai. Diduga penyebab penyakit adalah patogen tanah dan berdasarkan beberapa literatur penyebab penyakit ini adalah *Phytophthora sp* [11].



Gambar 2. 2 Penyakit Hawar Daun

2. Bercak Merah

Penyakit ini ditandai dengan terdapatnya bercak-bercak kecil-bulat berwarna merah keunguan pada daun tanaman stroberi. Penyakit ini jarang ditemukan pada pertanaman stroberi dengan tingkat kerusakan sangat rendah. Dari daun yang bergejala diisolasi dan dibiakkan di dalam Petridis, lalu diamati di bawah mikroskop. Dari pengamatan tersebut diamati konidia menyerupai *Pestalotia sp* [11].



Gambar 2. 3 Penyakit Bercak Merah Daun

3. Karat Daun

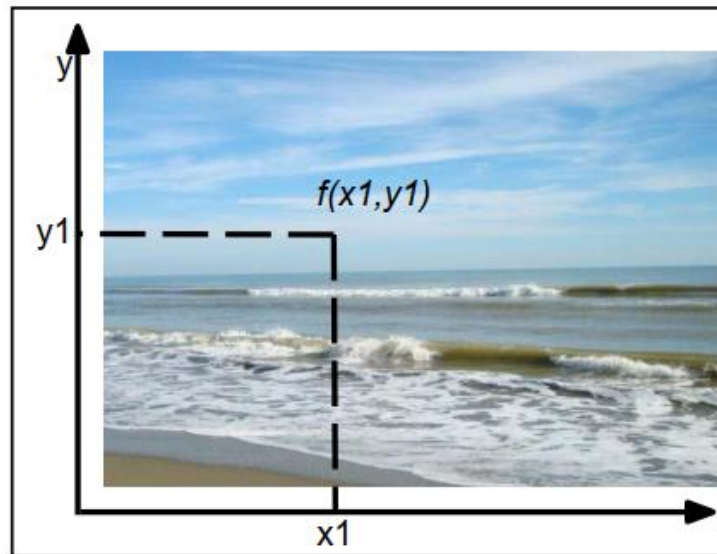
Gejala berupa bintik-bintik berwarna ungu yang berkembang menjadi coklat lalu disekitar bintik meluas daerah yang berwarna kekuningan hingga keunguan. Daun yang tua menjadi suram dan tumbuh merana kemudian mati (gugur). Penyakit ini cukup banyak ditemukan pada pertanaman stroberi yang tidak terawat dengan kondisi yang lembab [11].



Gambar 2. 4 Penyakit Karat Daun

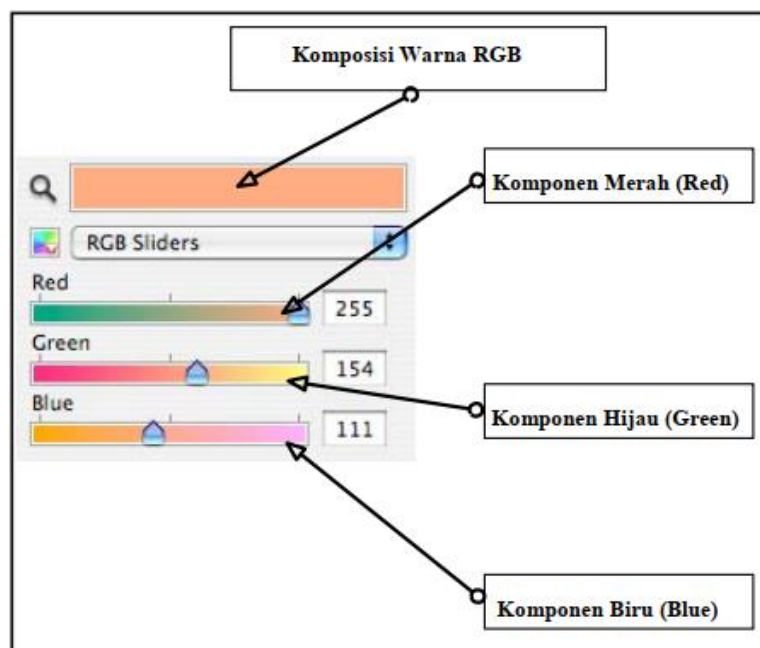
2.5 Pengolahan Citra Digital

Citra digital dapat didefinisikan sebagai fungsi dua variabel, $f(x,y)$, dimana x dan y adalah koordinat spasial dan nilai $f(x,y)$ adalah intensitas citra pada koordinat tersebut, hal tersebut diilustrasikan pada Gambar 2.5. Teknologi dasar untuk menciptakan dan menampilkan warna pada citra digital berdasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau dan biru (Red, Green, Blue - RGB). Komposisi warna RGB tersebut dapat dijelaskan pada Gambar 2.6.



Gambar 2. 5 Citra Digital

Sebuah citra diubah ke bentuk digital agar dapat disimpan dalam memori komputer atau media lain. Proses mengubah citra ke bentuk digital bisa dilakukan dengan beberapa perangkat, misalnya scanner, kamera digital dan handycam. Ketika sebuah citra sudah diubah ke dalam bentuk digital (selanjutnya disebut citra digital), bermacam-macam proses pengolahan citra dapat diperlakukan terhadap citra tersebut.



Gambar 2. 6 Komposisi Warna RGB

Operasi-operasi yang dilakukan di dalam pengolahan citra banyak ragamnya. Namun, secara umum, operasi pengolahan citra dapat diklasifikasikan dalam beberapa jenis sebagai berikut [12]:

1. Untuk memperbaiki tampilan citra (*image enhancement*).
2. Untuk mengurangi ukuran file dengan kualitas yang sama (*image compression*)
3. Untuk memulihkan citra ke kondisi semula (*image restoration*).
4. Untuk mengekstraksi ciri citra agar supaya lebih mudah dianalisis.

2.6 Klasifikasi K-Nearest Neighbor

K-Nearest Neighbor merupakan sebuah metode klasifikasi terhadap sekumpulan data berdasarkan pembelajaran data yang sudah terklasifikasi sebelumnya. KNN termasuk dalam golongan *supervised*, dimana hasil query *instance* yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam KNN. Nantinya kelas yang baru dari suatu data akan dipilih berdasarkan grup kelas yang dekat jarak vektornya[13].

Tujuan dari metode ini adalah mengklasifikasikan obyek baru berdasarkan atribut dan training sample. *Classifier* tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Diberikan titik *query*, akan ditemukan sejumlah k obyek atau (titik training) yang paling dekat dengan titik query. Klasifikasi menggunakan voting terbanyak diantara klasifikasi dari k obyek. Metode *k-nearest neighbor* (KNN) menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari query instance yang baru.

Metode *k-nearest neighbor* (KNN) sangatlah sederhana, bekerja berdasarkan jarak terpendek dari query instance ke training sample untuk menemukan KNN-nya. *Training sample* diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi training sample. sebuah titik pada ruang ini ditandai kelas c jika kelas c merupakan klasifikasi yang paling banyak ditemukan pada k buah tetangga terdekat dari titik tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan *Euclidean Distance*.

Jarak *Euclidian* paling sering digunakan menghitung jarak. Jarak *Euclidean* berfungsi menguji ukuran yang bisa digunakan sebagai interpretasi kedekatan jarak antara dua obyek. Yang direpresentasikan sebagai berikut:

$$d(\mathbf{x} - \mathbf{y}) = \sqrt{\sum_{j=1}^i (x_j - y_j)^2}$$

Keterangan:

d = jarak data uji ke data pembelajaran.

x_j = data uji ke- j , dengan $j = 1, 2, \dots, n$.

y_j = data pembelajaran ke- j dengan $j = 1, 2, \dots, n$

Nilai k yang terbaik untuk algoritma ini tergantung pada data. Secara umum, nilai k yang tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi semakin kabur. Nilai K yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan *cross-validation*. Kasus khusus dimana klasifikasi diprediksikan berdasarkan training data yang paling dekat (dengan kata lain, $k=1$) disebut metode nearest neighbor.

Ketepatan metode KNN sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap metode ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur agar performa klasifikasi menjadi lebih baik.

Langkah-langkah untuk menghitung k-nearest neighbor :

1. Menentukan parameter K (jumlah tetangga yang paling dekat).
2. Menghitung kuadrat jarak euclid (query instance) masing-masing obyek terhadap data sampel yang diberikan.
3. Kemudian mengurutkan objek-objek tersebut kedalam kelompok yang mempunyai jarak euclidean terkecil.
4. Mengumpulkan kategori Y (Klasifikasi nearest neighbor).

5. Dengan menggunakan kategori nearest neighbor yang paling mayoritas maka dapat diprediksikan nilai query instance yang telah dihitung.

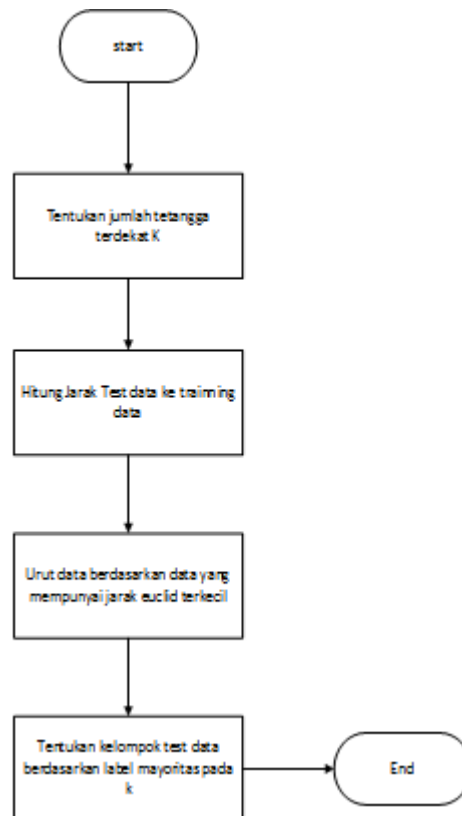
Kelebihan dari metode k-nearest neighbor adalah sebagai berikut:

- a. Ketangguhan terhadap data yang memiliki banyak noise.
- b. Efektif terhadap data yang berukuran besar.

Kekurangan dari metode k-nearest neighbor adalah sebagai berikut:

- a. Nilai k harus ditentukan secara manual.
- b. Training berdasarkan jarak harus menggunakan banyak sampel untuk mendapatkan hasil yang terbaik.
- c. Memerlukan komputasi tingkat tinggi karena perlunya menghitung satu persatu data testing terhadap semua data training.

Adapun metode dari KNN pada flowchart berikut :



Gambar 2. 7 Flow Chart metode KNN

2.7 Pendeteksian Tepi

Pendeteksian tepi merupakan teknik untuk menemukan garis tepi dari suatu objek pada citra dengan cara mendeteksi perubahan tingkat kecerahan yang signifikan atau memiliki diskontinuitas (*discontinuity*). Terdapatnya diskontinuitas lokal dalam nilai piksel yang melebihi ambang batas (*threshold*) yang diberikan disebut sebagai tepi (*edge*). Tepi juga bisa didefinisikan sebagai posisi citra dimana terjadi perubahan intensitas lokal yang terlihat jelas di sepanjang orientasi tertentu. Semakin besar perubahan intensitas lokal, maka semakin tinggi bukti yang menyatakan terdapat suatu tepi pada posisi tersebut [12].

Pendeteksi tepi dapat dilakukan dengan menggunakan highpass filter karena tepi termasuk ke dalam bagian citra berfrekuensi tinggi. Pendeteksian tepi bisa digunakan untuk segmentasi citra dan ekstraksi data untuk kebutuhan pengolahan citra, *computer vision*, dan *machine vision* [12].

2.8 K-Mean Clustering

Data Clustering merupakan salah satu metode Data Mining yang bersifat tanpa arahan (*unsupervised*). Ada dua jenis data clustering yang sering dipergunakan dalam proses pengelompokan data yaitu hierarchical (hirarki) data clustering dan non-hierarchical (non-hirarki) data clustering. K-Means merupakan salah satu metode data clustering non hirarki yang berusaha mempartisi data yang ada ke dalam bentuk satu atau lebih cluster/kelompok. Metode ini mempartisi data ke dalam cluster/kelompok sehingga data yang memiliki karakteristik yang sama dikelompokkan ke dalam satu cluster yang sama dan data yang mempunyai karakteristik yang berbeda dikelompokkan ke dalam kelompok yang lain. Adapun tujuan dari data clustering ini adalah untuk meminimalisasikan objective function yang diset dalam proses clustering, yang pada umumnya berusaha meminimalisasikan variasi di dalam suatu cluster dan memaksimalkan variasi antar cluster.

Data clustering menggunakan metode K-Means ini secara umum dilakukan dengan algoritma dasar sebagai berikut:

1. Menentukan pusat cluster

Pusat cluster digunakan untuk menentukan jarak data mana yang lebih dekat dan untuk mengelompokkan data dalam cluster.

2. Menghitung jarak data pada pusat cluster

Menghitung jarak pada pusat cluster dilakukan untuk mendapatkan jarak antara data pada pusat cluster yang terdekat.

3. Pengelompokan data

Pengelompokan data dilakukan dengan merubah jarak menjadi 0 dan 1. Hal ini dilakukan dengan cara merubah jarak terdekat menjadi nilai 1 dan jarak terjauh menjadi nilai 0.

4. Penentuan pusat cluster baru

Penentuan pusat cluster baru digunakan untuk menguji data yang dihasilkan tetap dan tidak berpindah pada pusat cluster lain. Penentuan pusat cluster baru dilakukan dengan cara membagi pusat cluster awal dengan jumlah anggotanya.

5. Pengulangan langkah kedua

Pengulangan langkah kedua dilakukan untuk memastikan posisi data tidak mengalami perubahan atau perpindahan pada cluster lain

2.9 OpenCV

OpenCV adalah suatu library gratis yang dikembangkan oleh developer-developer Intel Corporation. Library ini terdiri dari fungsi-fungsi computer vision dan API (Application Programming Interface) untuk image processing high level maupun low level dan sebagai optimasi aplikasi realtime. OpenCV sangat disarankan untuk programmer yang akan berkecukupan pada bidang computer vision, karena library ini mampu menciptakan aplikasi yang handal, kuat dibidang digital vision, dan mempunyai kemampuan yang mirip dengan cara pengolahan visual pada manusia,

Karena library ini bersifat cuma-cuma dan sifatnya yang open source, maka dari itu OpenCV tidak dipesan khusus untuk pengguna arsitektur Intel, tetapi dapat dibangun pada hampir semua arsitektur. Fitur-fitur yang dimiliki oleh OpenCV adalah :

1. Manipulasi data citra (*allocation, copying, setting, convert*).

2. Citra dan video I/O (file dan kamera *based input, image/video file output*).
3. Manipulasi Matriks dan Vektor beserta rutin-rutin aljabar linear (*products, solvers, eigenvalues, SVD*).
4. Data struktur dinamis (*lists, queues, sets, trees, graphs*).
5. Pemroses citra fundamental (*filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids*).
6. Analisis struktur (*connected components, contour processing, distance transform, various moments, template matching, hough transform, polygonal approximation, line fitting, ellipse fitting, delaunay triangulation*).
7. Kalibrasi kamera (*calibration patterns, estimasi fundamental matrix, estimasi homography, stereo correspondence*).
8. Analisis gerakan (*optical flow, segmentation, tracking*).
9. Pengenalan obyek (*eigen-methods, HMM*).
10. Graphical User Interface (*display image/video, penanganan keyboard dan mouse handling, scroll-bars*)

2.10 Internet Of Things (IoT)

Internet of Things atau dikenal dengan singkatan IoT, merupakan sebuah konsep dasar yang bertujuan untuk memperluas manfaat dari konektivitas internet yang tersambung secara terus-menerus. Adapun kemampuan seperti berbagi data, *remote control*, dan sebagainya, termasuk pada benda di dunia nyata. Contohnya bahan pangan, elektronik, koleksi, peralatan dan termasuk benda hidup yang segalanya tersambung ke jaringan lokal dan global melalui sensor yang tertanam dan selalu aktif [14].

Dalam kata *Internet of Things* terdapat kata “A Things” adalah sebagai subjek misalkan orang dengan monitor *implant* jantung. Hewan peternakan dengan transponder *biochip*, sebuah mobil yang telah dilengkapi oleh *built-in* sensor untuk memperingatkan pengemudi ketika tekanan ban rendah. *Internet of Things* paling erat hubungannya dengan komunikasi *machine-to-machine* (M2M) di bidang manufaktur dan listrik, perminyakan, dan gas. Produk dibangun dengan

kemampuan komunikasi M2M yang sering disebut dengan sistem cerdas atau "smart". (contoh: *smart farming*, *smart home*, *smart grid sensor*) [14].



Gambar 2. 8 Ilustrasi Internet of Things

Dapat kita simpulkan bahwa *internet of things* membuat suatu koneksi antara mesin dengan mesin, sehingga mesin-mesin tersebut dapat berinteraksi dan bekerja secara independen sesuai dengan data yang diperoleh dan diolahnya secara mandiri. Tujuannya adalah untuk membuat manusia berinteraksi dengan benda dengan lebih mudah, bahkan supaya benda juga bisa berkomunikasi dengan benda lainnya.

Teknologi *internet of things* sangat luar biasa. Jika sudah direalisasikan, teknologi ini tentu akan sangat memudahkan pekerjaan manusia. Manusia tidak akan perlu lagi mengatur mesin saat menggunakannya, tetapi mesin tersebut akan dapat mengatur dirinya sendiri dan berinteraksi dengan mesin lain yang dapat berkolaborasi dengannya. Hal ini membuat mesin-mesin tersebut dapat bekerja sendiri dan manusia dapat menikmati hasil kerja mesin-mesin tersebut tanpa harus repot-repot mengatur mereka.

Cara kerja dari *internet of things* cukup mudah. Setiap benda harus memiliki sebuah IP Address. IP Address adalah sebuah identitas dalam jaringan yang membuat benda tersebut bisa diperintahkan dari benda lain dalam jaringan yang sama. Selanjutnya, IP address dalam benda-benda tersebut akan dikoneksikan ke jaringan internet. Saat ini, koneksi internet sudah sangat mudah kita dapatkan.

Dengan demikian, kita dapat memantau benda tersebut bahkan memberi perintah kepada benda tersebut. Sebagai contoh, jika ada speaker yang memiliki IP address dan terkoneksi internet di Amerika Serikat, kita dapat memerintahkan speaker tersebut untuk menyalakan musik walaupun kita berada di Indonesia. Yang kita perlukan hanyalah koneksi internet. Lalu apa hubungannya dengan *internet of things*, Setelah sebuah benda memiliki IP address dan terkoneksi dengan internet, pada benda tersebut juga dipasang sebuah sensor. Sensor pada benda memungkinkan benda tersebut memperoleh informasi yang dibutuhkan. Setelah memperoleh informasi, benda tersebut dapat mengolah informasi itu sendiri, bahkan berkomunikasi dengan benda-benda lain yang memiliki IP address dan terkoneksi dengan internet juga. Akan terjadi pertukaran informasi dalam komunikasi antara benda-benda tersebut. Setelah pengolahan informasi selesai, benda tersebut dapat bekerja dengan sendirinya, atau bahkan memerintahkan benda lain juga untuk ikut bekerja. Jadi, dalam *internet of things* manusia akan bertindak sebagai raja dan akan dilayani oleh benda-benda disekitarnya [15].

2.11 Raspberry Pi 3 Model B

Raspberry Pi 3 merupakan generasi ketiga dari keluarga *raspberry pi*. *Raspberry pi 3 Model B+* adalah produk terbaru dalam jajaran seri *raspberry pi 3*, memiliki RAM 1 GB dengan chipset Broadcom BCM2837B0 Cortex A53 64-bit berkecepatan 1,4GHz. Chipset ini memiliki manajemen suhu yang lebih baik sehingga dapat berjalan pada kecepatan penuh dengan lebih lama sebelum mengalami throttling akibat panas. Perangkat ini menggunakan koneksi wireless dual band yang mendukung 802.11ac yang lebih kencang dibanding generasi sebelumnya serta dilengkapi juga dengan Bluetooth 4,2/BLE, jaringan Ethernet yang lebih cepat, dan kemampuan melakukan PoE melalui HAT PoE yang terpisah. *Raspberry Pi 3* juga memiliki 4 *USB port*, 40 *pin GPIO*, *Full HDMI port*, *Port Ethernet*, *Combined 3.5mm audio jack and composite video*, *Camera interface (CSI)*, *Display interface (DSI)*, slot kartu *Micro SD* (Sistem tekan-tarik, berbeda dari yang sebelumnya ditekan-tekan), dan *VideoCore IV 3D graphics core*.

LAN nirkabel dual-band hadir dengan sertifikasi penyesuaian modular, memungkinkan *board* dirancang untuk menjadi produk akhir dengan pengujian

kualitas LAN nirkabel yang berkurang secara signifikan, meningkatkan biaya dan waktu untuk memasarkan. *Raspberry Pi 3 Model B+* mempertahankan jejak mekanis yang sama seperti *Raspberry Pi 2 Model B* dan *Raspberry Pi 3 Model B* [21]. Pada penelitian ini *raspberry pi* digunakan sebagai mini pc untuk server website dan juga untuk pengolahan data yang didapat dari sensor melalui Arduino. Gambar adalah bentuk *raspberry pi 3 model b+* [16].



Gambar 2. 9 Raspberry Pi 3 Model B+

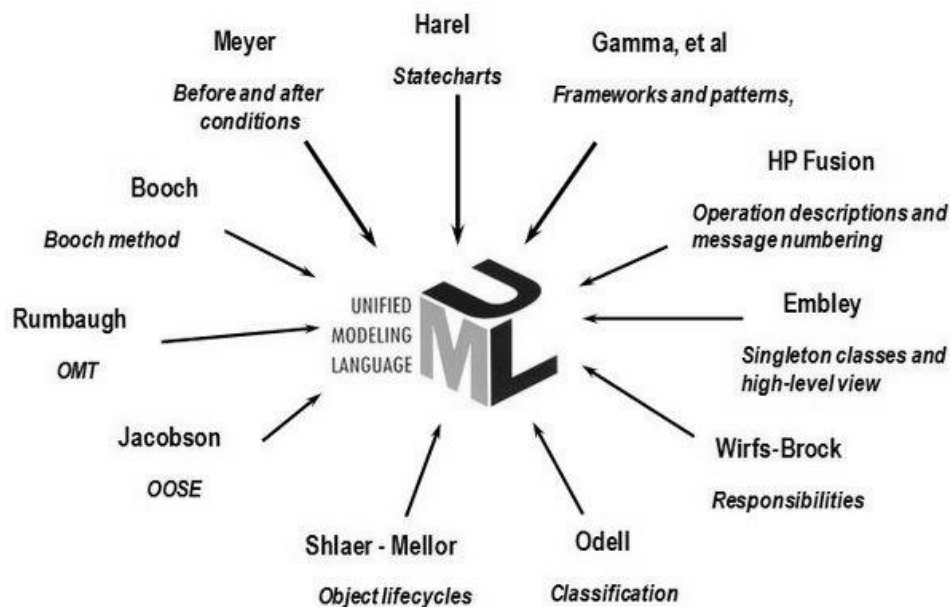
2.12 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa permodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atau svisi mekanisme yang efektif untuk berbagi dan mengkomunikasikan rancangan mereka dengan yang lain.

Unified Modeling Language (UML) merupakan kesatuan dari Bahasa permodelan yang dikembangkan booch, *Object Modeling Technique* (OMT) dan *Object Orented Software Engineering* (OOSE) [17]. Metode ini menjadikan proses analisis dan design kedalam empat tahapan iterative, yaitu identifikasi kelas-kelas, dan objek-objek, identifikasi semantic dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode Booch adalah pada detail dan kayanya dengan notasi dan elemen, permodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan entity-relationship. Tahapan utama dalam metodologi ini adalah nalisis, design sistem, design objek dan implementasi. Keunggulan metode ini adalah dalam

penotasian yang mendukung konsep OO. Metode OOSE dari Jacobson lebih memberi penekanan pada use case. OOSE memiliki tiga tahapan yaitu membuat model requirement dan analisis, design dan implementasi, dan model pengujian. Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam daripada metode lainnya []. Gambar 2.1 berikut adalah unsur-unsur yang membentuk UML.



Gambar 2. 10 Unsur-unsur pembentuk UML.

Sebagai sebuah notasi grafis yang relative sudah dibakukan (*open standard*) dan dikontrol oleh OMG (*Object Management Group*) mungkin lebih dikenal sebagai badan yang berhasil membakukan CORBA (Common Object Request Broker Architecture), UML menawarkan banyak keistimewaan. UML tidak hanya dominan dalam penotasian di lingkungan OO tetapi juga populer di luar lingkungan OO. Paling tidak ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan Bahasa pemograman. Sebagai sebuah sketsa, UML bisa berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dari sistem. Dengan demikian

semua anggota tim akan mempunyai gambaran yang sama tentang suatu sistem. UML bisa juga berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bisa diketahui informasi detail tentang coding program (*forward engineering*) atau bahkan membaca program dan menginterpretasikannya kembali kedalam diagram (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana code program yang tidak terdokumentasi asli hilang atau bahkan elum dibuat sama sekali. Sebagai bahasa pemrograman, UML dapat menterjemahkan diagram yang ada di UML menjadi code program yang siap untuk di jalankan [17].

2.12.1 Diagram UML

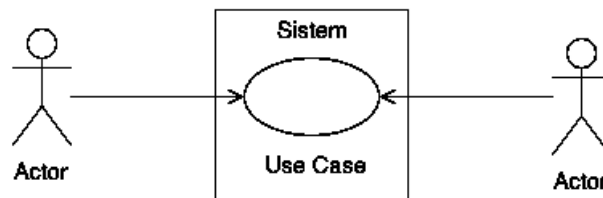
UML menyediakan berbagai macam diagram untuk memodelkan aplikasi perangkat lunak berorientasi objek [17]. Namun, pada penelitian ini hanya menggunakan 4 macam diagram saja untuk memodelkannya. Yaitu:

1. *Use Case Diagram*

Use case diagram adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara user (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras atau urutan waktu [17]. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan Bersama-sama oleh tujuan umum pengguna.

Use case adalah alat bantu terbaik guna menstimulasi pengguna potensial untuk mengatakan tentang suatu sistem dari sudut pandangnya. Tidak selalu mudah bagi pengguna untuk menyatakan bagaimana mereka bermaksud menggunakan sebuah sistem. Karena sistem pengembangan tradisional sering cerboh dalam melakukan analisis, akibatnya pengguna seringkali susah menjawabnya tatkala dimintai masukan tentang sesuatu [17].

Diagram use case menunjukkan 3 aspek dari sistem yaitu : *actor*, *use case* dan *sistem* atau *sub sistem boundary*. *Actor* mewakili peran orang, sistem yang lain atau alat ketika berkomunikasi dengan use case. Gambar 2.2 mengilustrasikan *actor*, *use case* dan *boundary*.

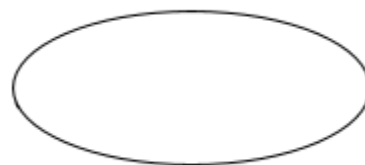


Gambar 2. 11 Use Case Model [17]

Berikut ini adalah bagian dari sebuah use case diagram :

a. *Use Case*

Use case adalah abstraksi dari interaksi antarsistem dengan *actor*. Oleh karena itu sangat penting untuk memilih abstraksi yang cocok. *Use case* dibuat berdasarkan keperluan *actor*. *Use case* harus merupakan ‘apa’ yang dikerjakan software aplikasi, bukan ‘bagaimana’ software aplikasi mengerjakannya. Setiap *user case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Nama *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case* yang memiliki nama yang sama. Gambar 2.3 menunjukkan bentuk *Use Case* dalam UML.

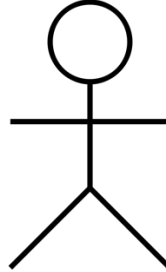


Gambar 2. 12 Use Case [17]

b. *Actors*

Actors adalah *abstraction* dari orang dan sistem yang lain yang mengaktifkan fungsi dari target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Bahwa actor berinteraksi dengan

use case, tetapi tidak memiliki control atas *use case*. Gambar 2.4 menunjukkan bentuk *actor* dalam UML.



Gambar 2. 13 Actor [17]

c. *Relationship*

Relationship adalah hubungan antara *use cases* dengan *actors*. *Relationship* dalam *use case* diagram meliputi:

a. Asosiasi antara *actor* dan *use case*.

Hubungan antara *actor* dan *use case* yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis lurus dari actor menuju *use case* baik dengan menggunakan mata panah terbuka ataupun tidak.

b. Asosiasi antara 2 *use case*

Hubungan antara *use case* yang satu dan *use case* lainnya yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis putus-putus/garis lurus dengan mata panah terbuka di ujungnya.

c. Generalisasi antara 2 *actor*

Hubungan *inheritance* (pewarisan) yang melibatkan *actor* yang satu (*the child*) dengan *actor* lainnya (*the parent*). Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.

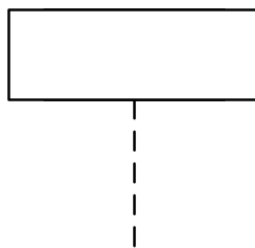
d. Generalisasi antara 2 *use case*.

Hubungan *inheritance* (pewarisan) yang melibatkan *use case* yang satu (*the child*) dengan *use case* lainnya (*the parent*). Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.

2. Sequence Diagram

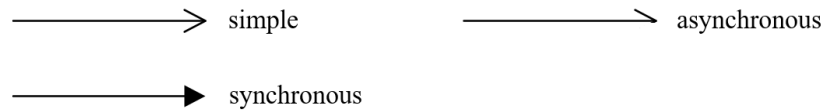
Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakan diantara obyek-obyek ini di dalam use case. Komponen utama *sequence diagram* terdiri atas objek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical* [17].

Objek diletakan di detak bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram, istilah objek dikenal juga dengan *participant*, setiap *participant* terhubung dengan garis titik-titik yang disebut *lifeline*. Sepanjang *lifeline* ada kotak yang disebut *activation*, *activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation* [17].



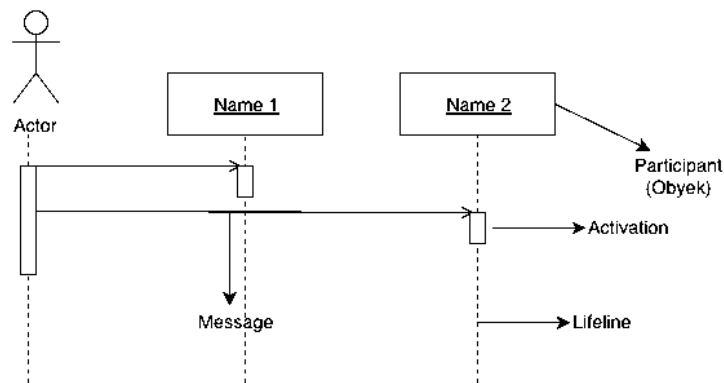
Gambar 2. 14 Participant pada sebuah sequence diagram [17]

Sebuah *message* bergerak dari satu *participant* ke *participant* yang lain dan dari satu *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri. Sebuah *message* bisa jadi simple, *synchronous* atau *asynchronous*. *Message* yang simple adalah sebuah perpindahan (*transfer*) *control* dari satu *participant* ke *participant* yang lainnya. Jika sebuah *participant* mengirimkan sebuah *message synchronous*, maka jawaban atas *message* tersebut akan ditunggu sebelum diproses dengan urursannya. Namun jika *message asynchronous* yang dikirimkan, maka jawaban atas *message* tersebut tidak perlu ditunggu.



Gambar 2. 15 Simbol-simbol message [17]

Time adalah diagram yang mewakili waktu pada arah *vertical*. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijadikan terlebih dahulu dibanding *message* yang lebih dekat ke bawah. Gambar 2.7 menunjukkan esensi *symbol* dari *sequence diagram* dan *symbol* kerjanya secara bersama-sama.



Gambar 2. 16 Simbol-simbol yang ada pada sequence diagram [17]

Berikut ini merupakan komponen dalam *sequence diagram* :

a. *Activations*

Activations menjelaskan tentang eksekusi dari fungsi yang dimiliki oleh suatu objek.

b. *Actor*

Actor menjelaskan tentang peran yang melakukan serangkaian aksi dalam suatu proses.

c. *Collaboration boundary*

Collaboration boundary menjelaskan tentang tempat untuk lingkungan percobaan dan digunakan untuk memonitor objek.

d. *Parallel vertical lines*

Parallel vertical lines menjelaskan tentang suatu garis proses yang menunjuk pada suatu *state*.

e. *Processes*

Processes menjelaskan tentang tindakan/aksi yang dilakukan oleh aktor dalam suatu waktu.

f. *Window*

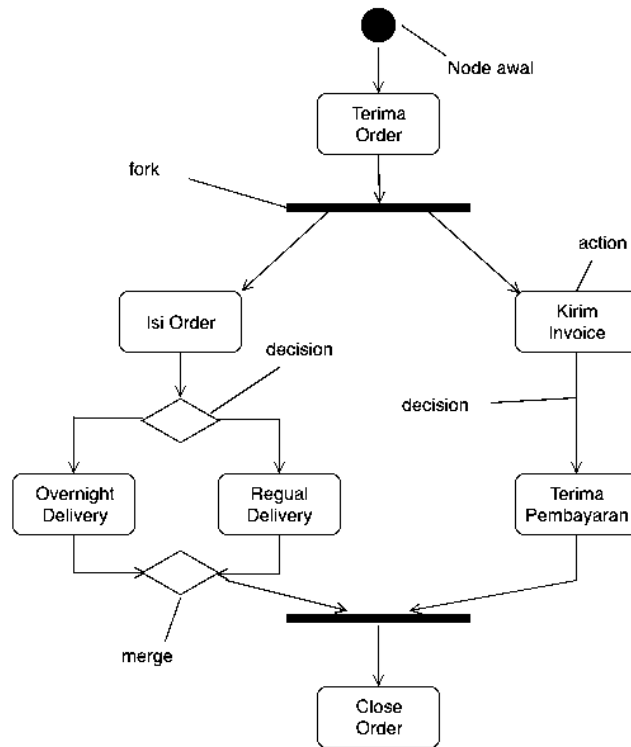
Window menjelaskan tentang halaman yang sedang ditampilkan dalam suatu proses.

g. *Loop*

Loop menjelaskan tentang model logika yang berpotensi untuk diulang beberapa kali.

3. *Activity Diagram*

Activity diagram adalah bagian penting dari UML yang menggambarkan aspek dinamis dari sistem. Logika procedural, proses bisnis dan aliran kerja suatu bisnis bisa dengan mudah dideskripsikan dalam *activity diagram*. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa. Tujuan dari *activity diagram* adalah untuk menangkap tingkah laku dinamis dari sistem dengan cara menunjukkan aliran pesan dari satu aktifitas ke aktifitas lainnya. Secara umum *activity diagram* digunakan untuk menggambarkan diagram alir yang terdiri dari banyak aktifitas dalam sistem dengan beberapa fungsi tambahan seperti percabangan, aliran paralel, swim lane, dan sebagainya. Sebelum menggambarkan sebuah *activity diagram*, perlu adanya pemahaman yang jelas tentang elemen yang akan digunakan di *activity diagram*. Elemen utama dalam *activity diagram* adalah aktifitas itu sendiri. Aktifitas adalah fungsi yang dilakukan oleh sistem. Setelah aktifitas teridentifikasi, selanjutnya yang perlu diketahui adalah bagaimana semua elemen tersebut berasosiasi dengan constraint dan kondisi. Langa selanjutnya perlu penjabaran tata letak dari keseluruhan aliran agar bisa ditransformasikan ke *activity diagram* [17]. Gambar 2.8 menunjukkan contoh *activity diagram* sederhana.



Gambar 2. 17 Contoh activity diagram sederhana [17]

Berikut ini merupakan komponen dalam activity diagram, yaitu :

a. *Activity node*

Activity node menggambarkan bentuk notasi dari beberapa proses yang beroperasi dalam kontrol dan nilai data

b. *Activity edge*

Activity edge menggambarkan bentuk *edge* yang menghubungkan aliran aksi secara langsung ,dimana menghubungkan *input* dan *output* dari aksi tersebut

c. *Initial state*

Bentuk lingkaran berisi penuh melambangkan awal dari suatu proses.

d. *Decision*

Bentuk wajib dengan suatu *flow* yang masuk beserta dua atau lebih *activity node* yang keluar. *Activity node* yang keluar ditandai untuk mengindikasikan beberapa kondisi.

e. *Fork*

Satu bar hitam dengan satu *activity node* yang masuk beserta dua atau lebih *activity node* yang keluar.

f. *Join*

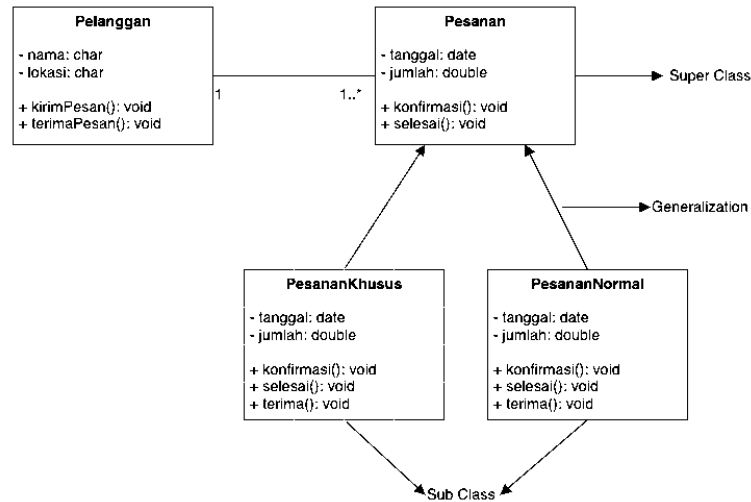
Satu bar hitam dengan dua atau lebih *activity node* yang masuk beserta satu *activity node* yang keluar, tercatat pada akhir dari proses secara bersamaan. Semua *actions* yang menuju *join* harus lengkap sebelum proses dapat berlanjut.

g. *Final state*

Bentuk lingkaran berisi penuh yang berada di dalam lingkaran kosong, menunjukkan akhir dari suatu proses.

4. *Class Diagram*

Class diagram adalah diagram statis. Ini mewakili pandangan statis dari suatu aplikasi. *Class diagram* tidak hanya digunakan untuk memvisualisasikan, menggambarkan, dan mendokumentasikan berbagai aspek sistem tetapi juga membangun kode eksekusi dari aplikasi perangkat lunak. *Class diagram* menggambarkan *atribut*, *operation* dan juga *constraint* yang terjadi pada sistem. *Class diagram* banyak digunakan dalam pemodelan sistem OO karena mereka adalah satu-satunya diagram UML, yang dapat dipetakan langsung dengan bahasa berorientasi objek. *Class diagram* menunjukkan koleksi *Class*, antarmuka, asosiasi, kolaborasi, dan *constraint*. Dikenal juga sebagai diagram structural [17]. Gambar 2.9 menunjukkan contoh *class diagram* sederhana.



Gambar 2. 18 Contoh class diagram sederhana [17]

Class diagram mempunyai 3 relasi dalam penggunaannya, yaitu :

a. *Assosiation*

Assosiation adalah sebuah hubungan yang menunjukkan adanya interaksi antar *class*. Hubungan ini dapat ditunjukkan dengan garis dengan mata panah terbuka di ujungnya yang mengindikasikan adanya aliran pesan dalam satu arah.

b. *Generalization*

Generalization adalah sebuah hubungan antar *class* yang bersifat dari khusus ke umum

c. *Constraint*

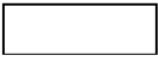



Constraint adalah sebuah hubungan yang digunakan dalam sistem untuk memberi batasan pada sistem sehingga didapat aspek yang tidak fungsional.

2.13 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh *Sistem Analys* dalam tahap analisis persyaratan proyek pengembangan sistem. Sementara seolah-olah teknik diagram atau alat peraga memberikan dasar untuk desain database relasional yang mendasari sistem informasi yang dikembangkan. ERD bersama-sama dengan detail pendukung merupakan model data yang pada

gilirannya digunakan sebagai spesifikasi untuk database [18]. Tabel 2.1 menunjukkan beberapa symbol dalam ERD.

Tabel 2. 2 Entity Relationship Diagram (ERD) [18]

No	Simbol	Keterangan
1		Entitas
2		Atribut
3		Hubungan
4		Garis

Penjelasan dari tabel adalah sebagai berikut [17] :

1. Entitas

Objek dalam dunia nyata yang dapat dibedakan dengan objek lain. Entitas terdiri atas beberapa atribut mengidentifikasi atau membedakan yang satu dengan yang lainnya. Pada setiap entitas baru harus memiliki 1 atribut unik atau yang disebut dengan *primary key*.

2. Atribut

Isi dari atribut mempunyai elemen yang dapat mengidentifikasi isi elemen satu dengan yang lain. Ada dua jenis atribut, yaitu:

- a. *Identifier (key)* digunakan untuk menentukan suatu *entity* secara unik (*primary key*).
- b. *Descriptor (nonkey attribute)* digunakan untuk menspesifikasi karakteristik dari suatu *entity* yang tidak unik.

3. Kardinalitas

Menyatakan jumlah himpunan relasi antar entitas. Pemetaan kardinalitas terdiri dari:

- a. *One-to-one*, sebuah entitas pada A berhubungan dengan entitas B paling banyak.

- b. *One-to-many*, sebuah entitas pada A berhubungan dengan entitas B lebih dari satu.
- c. *Many-to-many*, sebuah entitas pada A berhubungan dengan entitas B lebih dari satu dan entitas B berhubungan dengan entitas A lebih dari satu juga.

2.14 Website

Website merupakan fasilitas internet yang menghubungkan dokumen dalam lingkup local maupun jarak jauh. Dokumen pada *website* disebut dengan *web page* dan *link* dalam *website* memungkinkan pengguna bisa berpindah dari satu *page* ke *page* lain (*hyper text*), baik diantara *page* yang disimpan dalam *server* yang sama maupun *server* diseluruh dunia. *Pages* diakses dan dibaca melalui *browser* seperti Netscape Navigator, Internet Explorer, Mozilla Firefox, Google Chrome dan aplikasi *browser* lainnya [19].

Berdasarkan sifatnya, suatu *website* dibagi menjadi dua, yakni :

1. *Website Statis*

Website statis adalah *web* yang halamannya tidak berubah, biasanya untuk melakukan perubahan dilakukan secara manual dengan mengubah kode. *Website statis* informasinya merupakan informasi satu arah, yakni hanya berasal dari pemilik *software*-nya saja, hanya bisa diupdate oleh pemiliknya saja. Contoh *website statis* ini, yaitu profil perusahaan.

2. *Website Dinamis*

Website dinamis merupakan *web* yang halaman selalu *update*, biasanya terdapat halaman *backend* (halaman *administrator*) yang digunakan untuk menambah atau mengubah konten. *Web dinamis* membutuhkan database untuk menyimpan. *Website dinamis* mempunyai arus informasi dua arah, yakni berasal dari pengguna dan pemilik, sehingga peng-*update*-nya dapat dilakukan oleh pengguna dan juga pemilik website [19].

2.15 Database

Database adalah salah satu koleksi terorganisasi dari data terstruktur, yang disimpan dengan duplikasi item data yang minimum guna memberikan pool

(kelompok) data yang konsisten dan terkontrol. Data ini umum bagi semua sistem, namun independen terhadap program yang menggunakan data itu [19].

Database disimpan di dalam tabel, dan tabel mengandung data yang berhubungan, atau *entity*, seperti misalnya orang, produk, pesanan, dan sebagainya. Tujuannya adalah menjaga tabel tetap kecil dan dapat dikelola, serta *entity-entity* yang terpisah disimpan dalam tabel-tabel tersendiri. Tentu saja *entity* tidak dapat independen satu sama lain. Di dalam sebuah *database*, setiap tabel memiliki sebuah *field* yang memiliki nilai unik untuk setiap baris [20]. Dalam pengembangan sistem pengarsipan surat pada skripsi ini penulis menggunakan aplikasi *databases MySQL*, adapun beberapa penjelasan tentang *databases MySQL* sebagai berikut :

2.15.1 MySQL

MySQL adalah salah satu aplikasi sistem manajemen databases relasional yang handal dalam mengelolah *databases* yang sederhana maupun kompleks. *MySQL* mempunyai dua macam lisensi yang dikeluarkan oleh *MySQL AB*, suatu perusahaan Swedia, lisensi tersebut yaitu :

1. Open Source software : *MySQL* tersedia via GNU GPL (*General Public License*) untuk yang gratis.
2. Commercial License : tersedia bagi siapa saja yang menyukai GPL, jika ingin mengembangkan dan menggunakan *MySQL* sebagai bagian dari *software* produk baru maka pengembang harus membeli *license commercial* ini.

2.15.2 Keunggulan *MySQL*

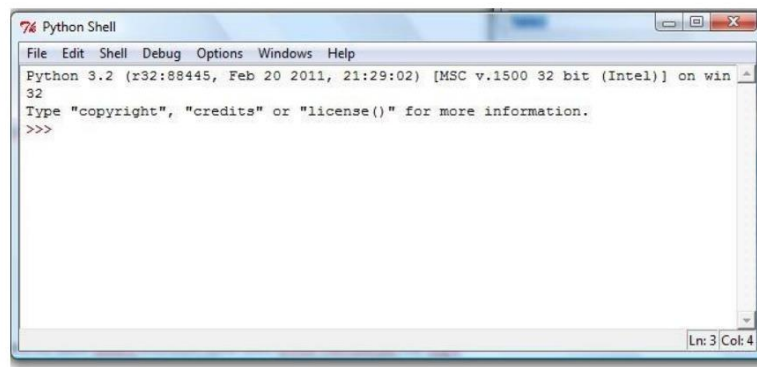
Dibawah ini beberapa keunggulan dari *databases MySQL* [16] :

1. Cepat : tujuan utama dari pengembangan *MySQL* adalah kecepatan dalam mengakses dan mengolah *databases*.
2. Tidak mahal : dibawah *Open Source software license* maka siapapun dapat menggunakan aplikasi *MySQL* secara gratis.
3. Mudah digunakan : kita dapat membangun dan berinteraksi dengan databases *MySQL* cukup dengan pernyataan sederhana didalam bahasa SQL.

4. Dapat berjalan pada beberapa sistem operasi : seperti Windows, Linux, Mac OS, Unix (solaris, AIX, DEC unix) FreeBSD, OS/2, Irix, dan lainnya.
5. Aman : MySQL adalah sistem otorisasi fleksibel yang memungkinkan beberapa atau semua privilege databases untuk pengguna khusus atau kelompok pengguna.

2.16 Python

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintak. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.



Gambar 2. 19 Tampilan Utama Python

Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya.

Dengan kode yang sederhana dan mudah diimplementasikan, seorang programmer dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari syntax error. Saat ini kode python dapat dijalankan di berbagai platform sistem operasi, beberapa di antaranya adalah [21].

- a. Linux/Unix
- b. Windows

- c. Mac OS X
- d. Java Virtual Machine
- e. OS/2
- f. Amiga
- g. Palm
- h. Symbian (untuk produk-produk Nokia)

2.17 Hypertext Preprocessor (PHP)

PHP menurut Anhar [22] adalah bahasa pemrograman web server-side yang bersifat open source, PHP juga merupakan script yang terintegrasi dengan HTML dan berada pada server (server side HTML embedded script). PHP juga merupakan script yang digunakan untuk membuat halaman website yang sangat dinamis, dinamis berarti halaman tampilan yang akan ditampilkan dibuat saat halaman itu diminta oleh client. PHP pertama kali dibuat oleh Rasmus Lerdorf seorang pemrogram C yang handal dari greenland Denmrak di tahun 1995, PHP diberi nama FI (Form Interpreted) yang digunakan untuk mengelola form dari web. Pada perkembangannya, kode-kode yang digunakan dirilis untuk umum sehingga mulai banyak dikembangkan oleh programmer diseluruh dunia. Tahun 1997 PHP dirilis dengan versi 2.0, pada versi ini sudah terintegrasi dengan bahasa pemrograman C dan sudah dilengkapi dengan modul sehingga kualitas kerja PHP lebih meningkat secara signifikan. Ditahun yang sama sebuah perusahaan program bernama Zend merilis ulang PHP versi ini dengan lebih baik, bersih dan cepat. Seiring berkembangnya jaman ditahun 1994 PHP versi 4.0 mulai dirilis dan versi ini paling banyak digunakan pada awal abad 21 karena PHP versi ini sudah mampu membangun web kompleks dengan stabilitas kecepatan yang tinggi. Ditahun 2004 perusahaan program Zend merilis PHP lagi dengan versi terbarunya 5.0 yang inti dari interpreter PHPH mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek kedalam PHP untuk menjawab perkembangan bahasa pemrograman kearah paradigma berorientasi objek.

Bahasa program PHP sering digunakan karena PHP adalah bahasa open source yang memiliki kesederhanaan dan memiliki beberapa fitur built-in yang berfungsi untuk menangani kebutuhan standart dalam pembuatan aplikasi web.

PHP juga merupakan bahasa script yang paling mudah dipahami karena memiliki beberapa referensi. PHP juga dapat digunakan untuk berbagai sistem operasi antara lain : Unix, Macintosh serta windows. PHP dapat dijalankan secara runtime melalui console serta dapat menjalankan perintah-perintah sistem. Open source disini memiliki arti code-code PHP terbuka untuk umum dan tidak berbayar atas pembelian dari license. Web server yang mendukung PHP dapat ditemukan dimana-mana, mulai dari Apache, IIS, Lighttpd hingga Xitami dengan konfigurasi yang relatif mudah. Selain itu PHP juga dilengkapi dengan berbagai macam pendukung lain seperti support langsung keberbagai macam databasea yang populer seperti Oracle, MySQL dan lain-lain.

2.18 Java Script Object Notation (JSON)

Java Script Object Notation (JSON) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll [23]. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

Penulis menggunakan metode JSON dalam pengiriman data yang dilakukan, karena JSON memiliki beberapa kelebihan - kelebihan dibandingkan XML, kelebihan – kelebihan tersebut adalah [23] :

1. Format Penulisan

Untuk merepresentasikan sebuah struktur data yang rumit dan berbentuk hirarkis penulisan JSON relatif lebih terstruktur dan mudah.

2. Ukuran

Ukuran karakter yang dibutuhkan JSON lebih kecil dibandingkan XML untuk data yang sama. Hal ini tentu berpengaruh pula pada kecepatan pertukaran data, walaupun tidak signifikan untuk data yang kecil, namun cukup berarti jika koneksi yang digunakan relatif lambat untuk mengakses aplikasi web kaya fitur yang memanfaatkan pertukaran data. Di sini JSON

lebih unggul dibandingkan XML, kecuali jika data dikompresi terlebih dahulu sebelum dikirimkan, perbedaan JSON dan XML yang telah dikompresi tidaklah signifikan.

3. Browser

Parsing Proses parsing merupakan proses pengenalan token atau bagian-bagian kecil dalam rangkaian dokumen XML/JSON. Contohnya, terdapat data text dalam format JSON. Data tersebut harus di-parsing terlebih dahulu sebelum dapat diakses dan dimanipulasi. Browser parsing berarti proses parsing yang terjadi pada sisi client/browser.

Melakukan browser parsing pada JSON lebih sederhana dibandingkan pada XML, JSON menggunakan function JavaScript eval() untuk melakukan parsing. Sementara dokumen XML di-parsing oleh XMLHttpRequest. Rata-rata survei menobatkan JSON sebagai pemenang jika diadu kecepatan parsingnya.

JSON terbuat dari dua struktur [23] :

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (object), rekaman (record), struktur (struct), kamus (dictionary), tabel hash (hash tabel), daftar berkunci (keyed list), atau associative array.
2. Daftar nilai terurutkan (an ordered list of values). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (array), vektor (vector), daftar (list), atau urutan (sequence).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemrograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan pada struktur data ini.

2.19 Metode Pengujian

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean. Sejumlah aturan yang berfungsi sebagai sasaran pengujian pada perangkat lunak adalah [24] :

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
2. Test case yang baik adalah test case yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

Karakteristik umum dari pengujian perangkat lunak adalah sebagai berikut

[24] :

1. Pengujian dimulai pada level modul dan bekerja keluar kearah integrasi pada sistem berbasis komputer.
2. Teknik pengujian yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya.
3. Pengujian diadakan oleh software developer dan untuk proyek yang besar oleh group testing yang independent.
4. Testing dan Debugging adalah aktivitas yang berbeda tetapi debugging harus diakomodasikan pada setiap strategi testing.

Metode pengujian perangkat lunak ada 3 jenis, yaitu [24] :

1. White Box/Glass Box - pengujian operasi.
2. *Black Box* - untuk menguji sistem.
3. Use case - untuk membuat input dalam perancangan *black box* dan pengujian statebased

2.19.1 White Box Testing

Pengujian *white box* adalah pengujian yang meramalkan cara kerja perangkat lunak secara rinci, karenanya logikal *path* (jalur logika) perangkat lunak akan di-test dengan menyediakan test case yang akan mengerjakan kumpulan kondisi atau pengulangan secara spesifik. Secara sekilas dapat diambil kesimpulan *white box* testing merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

Menurut Pressman [7], pengujian *White box* atau *Glass box* adalah metode *test case* desain yang menggunakan struktur kontrol desain *procedural* untuk

memperoleh *test-case*. Dengan menggunakan metode pengujian *white box*, analisis sistem akan dapat memperoleh *test case* yang:

- a. Memberikan jaminan bahwa semua jalur *independent* pada suatu modul telah digunakan paling tidak satu kali.
- b. Menggunakan semua keputusan logis dari sisi *true* dan *false*.
- c. Mengeksekusi semua batas fungsi *loops* dan batas operasionalnya.
- d. Menggunakan struktur *internal* untuk menjamin validitasnya.

Ujicoba basis *path* adalah teknik uji coba *white box* yang diusulkan Tom McCabe. Metode ini memungkinkan perancang *test case* mendapatkan ukuran kekompleksan logis dari perancangan prosedural dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. *Test case* yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.

Terdapat beberapa proses yang harus dilakukan dalam uji coba basis *path* yaitu diantaranya :

1. Notasi Diagram Alir

Sebelum metode basis *path* diperkenalkan, terlebih dahulu akan dijelaskan mengenai notasi sederhana dalam bentuk diagram alir (grafik alir). Diagram alir menggambarkan aliran kontrol logika yang menggunakan notasi.

2. Kompleksitas Siklomatis

Kompleksitas siklomatis adalah metrik perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila metrik ini digunakan dalam konteks metode pengujian basis *path*, maka nilai yang terhitung untuk kompleksitas siklomatis menentukan jumlah jalur independen dalam basis set suatu pemrograman memberi batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua statemen telah dieksekusi sedikitnya satu kali.

Jalur independen adalah jalur yang melalui program yang memperkenalkan sedikitnya satu rangkaian statemen proses baru atau suatu kondisi baru. Bila dinyatakan dengan terminologi grafik alir, jalur

independen harus bergerak sepanjang paling tidak satu *edge* yang tidak dilewatkan sebelum jalur tersebut ditentukan.

3. Melakukan Test Case

Metode uji coba basis *path* juga dapat diterapkan pada perancangan prosedural rinci atau program sumber. Pada bagian ini akan dijelaskan langkah-langkah uji coba basis *path*.

4. Matriks Grafis

Prosedur untuk mendapatkan grafik alir dan menentukan serangkaian basis *path*, cocok dengan mekanisasi. Untuk mengembangkan peranti perangkat lunak yang membantu pengujian basis *path*, struktur data yang disebut matriks grafis dapat sangat berguna.

Matriks grafis adalah matriks bujur sangkar yang ukurannya sama dengan jumlah simpul pada grafik alir. Masing-masing baris dan kolom sesuai dengan simpul yang diidentifikasi dan *entry* matriks sesuai dengan *edge* diantara simpul.

2.19.2 Black Box Testing

Pengujian menggunakan sekumpulan aktifitas validasi, dengan pendekatan *black box* testing. 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 bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian *black box* testing harus dibuat dengan kasus benar dan kasus salah.

Menurut Pressman [7], *black box* testing juga disebut pengujian tingkah laku, memusat pada kebutuhan fungsional perangkat lunak. Teknik pengujian *black box* memungkinkan memperoleh serangkaian kondisi masukan yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Beberapa jenis kesalahan yang dapat diidentifikasi adalah fungsi tidak benar atau hilang, kesalahan antar muka, kesalahan pada struktur data (pengaksesan basis data), kesalahan performansi, kesalahan inisialisasi dan akhir program.

Equivalence Partitioning merupakan metode *black box testing* yang membagi domain masukan dari program kedalam kelas-kelas sehingga test case dapat diperoleh. Equivalence Partitioning berusaha untuk mendefinisikan kasus uji yang menemukan sejumlah jenis kesalahan, dan mengurangi jumlah kasus uji yang harus dibuat. Kasus uji yang didesain untuk Equivalence Partitioning berdasarkan pada evaluasi dari kelas ekuivalensi untuk kondisi masukan yang menggambarkan kumpulan keadaan yang valid atau tidak. Kondisi masukan dapat berupa spesifikasi nilai numerik, kisaran nilai, kumpulan nilai yang berhubungan atau kondisi boolean.

Kesetaraan kelas dapat didefinisikan menurut panduan berikut [7] :

1. Jika masukan kondisi menentukan kisaran, satu sah dan dua diartikan tidak valid kesetaraan kelas.
2. Jika masukan membutuhkan nilai, kondisi tertentu satu sah dan dua tidak valid kesetaraan kelas diartikan.
3. Jika masukan kondisi menentukan anggota dari set, satu sah dan satu tidak valid kesetaraan kelas diartikan.
4. Jika kondisi yang input, boolean satu sah dan satu tidak valid kelas diartikan.

Menerapkan pedoman untuk derivasi kelas kesetaraan, uji kasus untuk setiap masukan domain item data dapat dikembangkan dan dilaksanakan. Uji kasus dipilih sehingga jumlah terbesar dari atribut dari kelas kesetaraan tersebut dilakukan sekaligus.

Beberapa kata kunci dalam pengujian perangkat lunak yang dapat diperhatikan, yaitu [25] :

1. Dinamis

Pengujian perangkat lunak dilakukan pada masukan yang bervariasi. Masukan ini ditentukan sebelum pengujian dilakukan dengan batasan yang disesuaikan dengan kemampuan perangkat lunak. Masukan tidak harus sesuatu yang dimungkinkan terjadi pada penggunaan program lebih lanjut, melainkan meliputi keseluruhan batasan yang dapat dijangkau perangkat lunak dan dilakukan pemercontohan (*sampling*) secara acak untuk proses pengujian.

2. Terbatas

Meskipun pengujian dilakukan pada perangkat lunak sederhana sehingga rumit sekalipun, pengujian dilakukan dengan memenuhi batasan-batasan tertentu sesuai dengan kemampuan program. Batasan ini juga diberlakukan pada masukan-masukan yang dipilih untuk pengujian. Tidak semua kemungkinan masukan diujikan pada perangkat lunak karena akan memakan waktu yang cukup panjang mengingat begitu banyaknya kemungkinan yang bisa terjadi. Untuk mengatasi hal ini, pemilihan masukan-masukan pada proses pengujian secara acak yang diperkirakan mampu memenuhi kebutuhan pengujian perangkat lunak akan dilakukan.

3. Tertentu

Pengujian dilakukan dengan batasan tertentu disesuaikan dengan harapan pada fungsi, respon, dan karakteristik perangkat lunak tersebut. Batasan tersebut akan disesuaikan dengan teknik-teknik pengujian yang ada. Pemilihan kriteria pengujian yang paling tepat merupakan hal yang kompleks. Dalam praktiknya, analisis risiko pengujian dan pengalaman terhadap pengujian-pengujian sejenis akan diperlukan.

4. Harapan

Kata kunci ini memiliki keadaan-keadaan yang diharapkan, baik berupa respon sistem terhadap masukan maupun karakteristik responnya. Dalam hal ini, batasan-batasan hasil pengujian yang diharapkan harus ditentukan. Dengan demikian, dapat diketahui apakah perangkat lunak tersebut telah memenuhi hasil pengujian yang diharapkan atau memerlukan pembenahan kembali, baik berupa perbaikan maupun pengembangan perangkat lunak.