# BAB III TINJAUAN PUSTAKA

## Penelitian Terkait

Dalam penelitian terdahulu ini diharapkan peneliti dapat melihat perbedaan antara penelitian yang telah dilakukan dengan penelitian sekarang. Selain itu, juga diharapkan dalam penelitian ini dapat diperhatikan mengenai kekurangan dan kelebihan antara penelitian terdahulu dengan penelitian sekarang.

Lutfi. Ahmad (2017) dengan judul “sistem informasi penjualan berbasis website pada kedai kopi ABC”.

Masalah yang diambil dalam penelitian ini adalah proses digitalisasi kegiatan penjualan kopi, untuk efisiensi penjualan, dan informasi laba rugi. Untuk menyelesaikan masalah tersebut, peneliti membuat sebuah sistem informasi penjualan yang berbasis website. Diamana sistem ini memiliki fitur informasi penjualan dan laba rugi. Kekurangan dalam penelitian ini adalah tidak adanya informasi persediaan barang yang bisa menyebabkan informasi laba rugi menjadi kurang akurat.

Habib. Muhsin (2021) dengan judul “rancang bangun sistem informasi penjualan café berbasis website pada UD. Gaharu Robotic Carwash”

Masalah yang diambil dalam penelitian ini adalah proses bisnis yang masih manual seperti pencatatan penjualan, pencatatan barang, dan penggajian. Untuk menangani masalah tersebut penulis membuat sistem informasi penjualan yang dapat memberikan informasi penjualan, dan barang. Akan tetapi sistem ini belum online sehingga untuk pemilik harus datang ke lokasi untuk memeriksa laporan.

Saputro. Wahyu tri (2021) “sistem informasi penjualan pada cranberrieskopi berbasis web”

Masalah dalam penelitian ini adalah proses transaksi yang masih manual, dimana penjual dan pembeli bertemu di toko. Untuk mengatasi masalah tersebut peneliti membuat sistem informasi penjualan diamana sistem ini bisa mengatasi masalah penjualan melalui aplikasi tanpa perlu bertemu antara penjual dan pembeli, pembeli bisa langsung memesan melalui aplikasi tersebut dan penjual akan mengantarkanya ke rumah pembeli. Kelemahan dari penelitian ini hanya membahas proses transaksi.

Murdianty, Agustina , Christy veronica (2014) “sistem informasi penjualan pada coffee shop studi kasus: Krakatoa coffe and gemstone”

Masalah yang di ambil peneliti adalah proses bisnis yang masih manual dengan melakukan pencatatan pesanan ke dalam kertas dan mengumpulkanya sebagai laporan. Solusi untuk masalah tersebut adalah dengan membuat sistem informasi penjualan yang meyediakan informasi penjualan dan keuntungan untuk pemilik usaha. Akan tetapi sistem ini memiliki kelemahan dengan tidak adanya sistem informasi persediaan yang akan mengakibatkan informasi keuntungan akan menjadi tidak akurat.

Imelda. Rijal ST.,MT (2019) “sistem informasi produksi pada kedai kopi cocoffe-in”

Masalah yang di ambil peneliti adalah proses bisnis yang masih manual dengan melakukan pencatatan pesanan dan persediaan barang kedalam buku hal ini mengakibatkan ketidak akuratan informasi data. Solusi untuk masalah tersebut adalah dengan membuat sistem informasi produksi kopi yang memberikan informasi transaksi dan ketersediaan barang. Kekurangan dalam penelitian ini adalah sistem ini masih berjalan di lokal.

## Pengertian Coffee shop

Cafe atau *Coffee Shop* (kedai kopi) adalah suatu tempat (kedai) yang menyajikan olahan kopiespreso dan kudapan kecil. Seiring perkembangan jaman coffe shop menyediakan makan kecil dan makanan berat. (Sumber: Kamus Besar Bahasa Indonesia, Departement Pendidikan dan Kebudayaan,Jakarta: Balai Pustaka, 1988)

## Definisi Sistem

Secara etimologi, Sistem berasal dari bahasa Latin (systcma) dan bahasa Yunani (sustcma) yang berarti suatu kesatuan yang terdiri komponen atau elemen yang dihubungkan bersama untuk memudahkan aliran informasi, materi atau energi untuk mencapai tujuan (A. S. Nugroho, 2017).

(Jogiyanto, 2018) mendefinisikan sistem sebagai gabungan dari berbagai elemen yang berhubungan dan berinteraksi untuk mencapai tujuan tertentu. system ini menggambarkan kejadian-kejadian dan kesatuan objek nyata, misalnya tempat, benda, dan orang-orang yang benar-benar ada dan nyata. Sehingga secara umum sistem diartikan sebagai suatu kumpulan objek atau unsur-unsur atau bagian-bagian yang memiliki arti berbeda-beda yang saling memiliki hubungan, saling berkerjasama dan saling mempengaruhi satu sama lain serta memiliki keterikatan pada rencana yang sama dalam mencapai suatu tujuan tertentu pada lingkungan yang kompleks.

## Definsi Informasi

Menurut I Putu Agus Eka Pratama (2014), Informasi merupakan hasil pengolahan data dari satu atau berbagai sumber, yang kemudian diolah, sehingga memberi nilai, arti, dan manfaat.

Menurut lukman ahmad dan munawir (2018) ciri – ciri informasi yang berkualitas adalah sebagai berikut:

Accessibility, Informasi harus mudah diakses, ada /tersedia, semakin mudah dan semakin banyak informasi makan akan semakin baik.

Timelines, Tepat waktu, informasi yang terlambat akan berakibat tidak baik, informasi yang baik harus cepat.

Relevance, Informasi yang dihasilkan relevan dan sesuai dengan kebutuhan organisasi /perusahaan atau orang yang membutuhkannya.

Accuracy, Informasi harus tepat, akuran, bebas dari kesalahan.

Precision, Informasi harus presisi atau terperinci /detail.

Useful, Informasi yang bermanfaat, memiliki nilai kegunaan.

## Definisi Manajemen

Managemen disebut sebagai seni untuk merealisasikan pekerjaan melalui orang lain dalam kaitannya dengan organisasi bahwa manager dalam mencapai tujuan organisasi dengan cara pengaturan orang lain (pegawai). Manajemen mencakup fungsi perencanaan, pengorganisasian, penyusunan personalia, pengarahan dan pengawasan (I Heryanto, 2016).

Menurut George R. Terry (Ahmad, 2018) manajemen merupakan proses yang khas yang terdiri dari tindakan-tindakan perencanaan, pengorganisasian, penggerakkan, dan pengawasan untuk mencapai tujuan yang telah ditetapkan memalui pemanfaatan sumber daya manusia dan sumber daya lainnya Definisi lain Stoner AF (Ahmad, 2018) menyatakan bahwa manajemen merupakan proses perencanaan, pengorganisasian dan pengawasan antar anggota organisasi dengan menggunakan seluruh sumber

## Definisi Sistem Manajemen

Definisi Sistem Managemen menurut (Ahmad, 2018) sistem managemen adalah jaringan prosedur pengolahan data yang dikembangkan dalam organisasi dan disatukan apabila di pandang perlu, dengan maksud memberikan data kepada manajemen setiap waktu diperlukan, baik data yang bersifat intern maupun yang bersifat ekstern, untuk dasar pengambilan keputusan dalam rangka mencapai tujuan organisasi.

## Definisi Stok Barang

Menurut Kotler dalam Utama (2012) ketersediaan barang adalah kemampuan perusahaan untuk menjaga persediaan produk ketika terjadi peningkatan permintaan terhadap merek produk.

Menurut Xu et al dalam Emiri (2011) menyatakan bahwa ketersediaan merupakan faktor ketertarikan berdasarkan logika atau pertimbangan-pertimbangan bagaimana barang mudah diperoleh.

## Definisi Website

Menurut Bekti (2015) Website merupakan kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara,dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masingmasing dihubungkan dengan jaringan-jaringan halaman.

Menurut Rahmadi (2013) ”website (lebih dikenal dengan sebutan situs) adalah sejumlah halaman web yang memiliki topik saling terkait, terkadang disertai pula dengan berkas-berkas gambar, video atau jenis-jenis berkas lainnya.”.

Sedangkan menurut Ippho Santoso dalam Rahmadi (2013) “membagi website menjadi golongan kanan dan golongan kiri. Dalam website dikenal dengan sebutan website dinamis dan website statis.

Website statis

Website statis adalah website yang mempunyai halaman konten yang tidak berubah-ubah.

Website dinamis

Website dinamis merupakan website yang secara struktur ditujukan untuk update sesering mungkin. Dari uraian teori diatas penulis menarik kesimpulan website adalah kumpulan halaman-halaman yang dapat menampilkan teks, gambar, animasi, video, suara yang masing-masing dihubungkan dengan jaringan-jaringan halaman. Website dibagi menjadi dua golongan yaitu website statis dan website dinamis.

## Definisi MySQL

MySQL adalah sebuah implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersil. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basis data yang telah ada sebelumnya SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. MySQL memiliki beberapa keistimewaan, antara lain (B. Nugroho, 2019):

*Portabilitas.* MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Max Os X Server, Solaris, Amiga, dan masih banyak lagi.

Perangkat lunak sumber terbuka. MySQL didistribusikan sebagai perangkat lunak sumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis.

*Multi-user*. MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah.

* + 1. *Performance-tuning*.

MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL persatuan waktu.

* + 1. Ragam tipe data.

MySQL memiliki ragam tipe data yang sangat kaya, seperti signed/unsigned integer, float, double, char, text, date, timestamp, dan lain-lain.

* + 1. Perintah dan fungsi.

MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *select* dan *where* dalam perintah (*query*).

* + 1. Keamanan.

MySQL memiliki beberapa lapisan keamanan seperti level *subnetmask*, nama *host*, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenskripsi.

* + 1. Skalabilitas dan pembatasan.

MySQL mampu menangani basis data secara besar, dengan jumlah rekaman (*records*) lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

* + 1. Konektivitas.

MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, UNIX soket (UNIX), atau Named Pipes (NT).

* + 1. Lokalisasi.

MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meskipun demikian, Bahasa Indonesia belum termasuk di dalamnya.

* + 1. Antar muka.

MySQL memiliki antar muka (*interface*) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).

* + 1. Klien dan peralatan

MySQL dilengkapi dengan berbagai peralatan (*tool*) yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk *online*.

* + 1. Struktur tabel

MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan dengan basis data lainnya semacam PostgreSQL ataupun Oracle.

## Pengertian PHP (Hypertext Prepocessor)

Pada awalnya PHP merupakan kependekan dari *Personal Home Page*. PHP pertama kali dibua oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP masih bernama *Form Interpreted* (FI), yang wujudnya masih berupa sekumpulan skrip yang digunakan untuk mengolah data formulir dari web. Selanjutnya Rasmus merilis kode sumber untuk umum dan menamakannya PHP/FI. Dengan perilisan kode sumber ini menjadi sumber terbuka, maka banyak pemrogram yang tertarik untuk ikut mengembangkan PHP. PHP memiliki 8 (delapan) tipe data, yaitu (Abdulloh, 2018) :

1. *Integer*

Tipe ini meliputi semua bilangan bulat dengan range -2,147,483,648 sampai +2,147,483,647 pada *platform* 32bit. PHP juga akan mengkonversi secara otomatis bila suatu bilangan berada diluar *range* tersebut ke dalam tipe data *floating point*. Tipe ini juga dapat dinyatakan dalam bentuk oktal (berbasis 8), desimal (berbasis 10), heksadesimal (berbasis 16).

1. *Floating Point*

Tipe ini biasa digunakan dalam bilangan pecahan namun bisa juga bilangan desimal. Tipe ini memiliki range 1.7E-308 sampai 1.7E+308. Dapat dinyatakan dalam bentuk bilangan desimal atau dalam bentuk pangkat.

1. *String*

Tipe data string dinyatakan dengan mengapitnya menggunakan tanda petik tunggal (‘ ‘) atau tanda petik ganda (” “). Perbedaan dari penggunaan keduanya adalah dengan tanda petik tunggal kita tidak dapat menggunakan variable dan *escape sequence handling* bersama dalam suatu kalimat.

1. Boolean

Tipe data boolean digunakan untuk menyimpan nilai *true* atau *false.* Biasanya tipe data ini mayoritas digunakan untuk melakukan pengecekan kondisi pada php.

1. Null

Tipe data yang tidak memuat apapun. Setiap Variabel yang diset menjadi Tipe Data NULL ini akan menjadikan Variabel tersebut kosong.

1. *Array*

Tipe ini dapat mengandung satu atau lebih data juga dapat diindeks berdasarkan numerik atau string. Ia juga mendukung multiarray dimensi dan membolehkan semua datanya berbeda tipe data.

1. Object

Tipe data object dapat berupa bilangan, variable atau fungsi. Object dibuat dengan tujuan agar para programmer terbiasa dengan OOP, meski fasilitas ini masih minim.

1. *Resource*

Tipe Data Spesial yang satu ini di khususkan untuk menyimpan resource, sumber atau alamat. Variabel tersebut hanya dapat diciptakan oleh suatu fungsi khusus yang mengembalikan nilai berupa resource seperti penggunaan fungsi fopen, opendir, mysql\_connect, mysql\_query dan semacamnya.

Beberapa kelebihan PHP dari bahasa pemrograman *web*, antara lain :

1. Bahasa pemrograman PHP adalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaannya.
2. *Web* Server yang mendukung PHP dapat ditemukan dimana-mana, mulai dari apache, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis-milis dan developer yang siap membantu dalam pengembangan.
4. Dalam sisi pemahaman, PHP adalah bahasa *scripting* yang paling mudah karena memiliki referensi yang banyak.

PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara *runtime* melalui *console* serta juga dapat menjalankan perintah-perintah sistem.

## Definisi Framework

Dalam penelitian ini penulis akan mengimplementasikan sistem menggunakan framework Laravel. Dimana framework ini akan mempermudah prosses pembuatan aplikasi dan pembacaan kode program

Menurut supono (2016) framework secara sederhana dapat diartikan kumpulan dari fungsi – fungsi / prosedur – prosedur dan class - class untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang programmer tanpa harus membuat fungsi atau class dari awal.

## Pengenalan UML (Unified Modelling Language)

Pada perkembangan teknik pemograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemograman berorientasi objek yaitu Unified Modelling Language (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung (Muslihudin, 2016).

UML hanya berfungsi sebagai untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

### Pengertian usecase diagaram

*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, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu (Muslihudin, 2016).

*Use case diagram* dapat sangat membantu bila sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. *Sebuah* *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. Berikut adalah simbol - simbol *Use Case Diagram*.

Tabel 3. 1. Simbol Use case Diagram (Muslihudin, 2016)

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | *Actor* | Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*. |

Tabel 3.1. Simbol Use case Diagram (Muslihudin, 2016) (lanjutan)

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*). |
|  | *Generalization* | Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
|  | *Include* | Menspesifikasikan bahwa *use case* sumber secara *eksplisit*. |
|  | *Extend* | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan. |
|  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
|  | *System* | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
|  | *Use Case* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |

Tabel 3.1. Simbol Use case Diagram (Muslihudin, 2016) (lanjutan)

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |
|  | *Note* | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi |

### Pengertian 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). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain (Muslihudin, 2016). *Class* memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

*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*. Hubunganantar *class* yaitu:

* 1. Asosiasi

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.

* 1. Agregasi

Hubungan yang menyatakan bagian (“terdiri atas..”).

* 1. Pewarisan

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.

* 1. Hubungan Dinamis

Rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain.

Tabel 3. 2.Simbol Class Diagram (Muslihudin, 2016)

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | *Generalization* | Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
|  | *Nary Association* | Upaya untuk menghindari asosiasi dengan > 2 objek. |
|  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |

Tabel 3.2. Simbol Class Diagram (Muslihudin, 2016) (lanjutan)

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
|  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |
|  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri |
|  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya |

### Pengertian 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 (Muslihudin, 2016).

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. Berikut adalah simbol - simbol Activity Diagram.

Tabel 3. 3. Simbol Activity Diagram (Muslihudin, 2016)

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | State | Kondisi yang mungkin dialami oleh suatu obyek. |
|  | Note | Note digunakan untuk memberikan keterangan ataukomentar |
|  | Aktivitas | Perilaku obyek yang dilakukan saat obyek berada dalam *state* tertentu. |
|  | Start State | *Start state* digunakan untuk memulai diagram *statechart.* |
|  | End State | End start digunakan untuk mengakhiri diagram. |

Tabel 3.3. Simbol Activity Diagram (Muslihudin, 2016) (lanjutan)

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | Decision | *Decision* digunakan sebagai pilihan untuk pengambilan keputusan. |
|  | Penggabungan / Join | Digunakan untuk split dan join.pada saat diagram akanmembagi 2, bar ini akanditambahkan. dan sebelum diagram digabung menjadi satu, sebagai join. |
|  | Asosiasi *(association)* | Apa yang menghubungkan antara objek satu dengan objek yang lainnya. |

### Pengertian 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). *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 men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan (Muslihudin, 2016).

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*. Berikut adalah simbol - simbol *Sequence Diagram*.

Tabel 3. 4. Simbol Sequence Diagram (Muslihudin, 2016)

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | *LifeLine* | Menggambarkan objek *entity*, antarmuka yang saling berinteraksi. |
|  | *Message* | Menggambarkan pengiriman pesan |
|  | *Actor* | Menggambarkan orang yang sedang berinteraksi dengan sistem |
|  | *Entity Class* | Menggambarkan hubungan kegiatan yang akan dilakukan |
|  | *Boundary Class* | Menggambarkan sebuah penggambaran dari form |
|  | *Control Class* | Menggambarkan penghubung antara *boundary* dengan tabel |

## Pengujian sistem

“pengujian adalah proses menjalankan perangkat lunak dengan tujuan untuk menemukan (dan akhirnya membetulkan) kesalahan.” (Roger S. pressman, 2012).

Pengujian adalah satu set aktivitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan. (Rosa A.S., M. Shalahuddin, 2018). Pengujian untuk validasi memiliki beberapa pendekatan sebagai berikut:

### Pengujian black box

Pengujian black box Yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. (Rosa A.S., M. Shalahuddin, 2014).

Black-Box Testing berfokus pada persyaratan fungsional perangkat lunak, dengan demikian memungkinkan engineer untuk memperoleh input yang akan melaksanakan persyaratan fungsional sebuah program.

Black-Box Testing dapat menemukan kesalahan dalam kategori berikut (Pressman, 2012):

fungsi yang tidak benar atau hilang.

Kesalahan pada tampilan.

Kesalahan perilaku kinerja

Kesalahan dalam struktur data atau akses database eksternal

Kesalahan inisialisasi dan penghentian

### Pengujian white box

Pengujian *whitebox* yaitu menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi kebutuhan. Pengujian kotak putih dilakukan dengan memeriksa logika dari kode program. (Rosa A.S., M. Shalahuddin, 2014).

Metode pengujian jalur-jalur dapat diterapkan untuk perancangan prosedural atau kode program. Meskipun average merupakan algoritme yang sangat sederhana, namun memuat di dalamnya kondisi gabungan dan perulangan. Beberapa langkah berikut ini dapat di terapkan untuk menurunkan basis set:

Menggunakan perancangan atau kode sebagai sebuah dasar dengan membuat gambar grafik alir yang sesuai. Sebuah grafik alir dibuat dengan menggunakan simbol dan aturan-aturan konstruksi dengan memberi nomor yang akan dipetakan ke node grafik alir yang sesuai.

Menentukan kompleksitas siklomatik dari alir grafik yang dihasilkan. Kompleksitas siklomatik V(G) ditentukan dengan menerapkan algoritma yang ada. V(G) dapat ditentukan dengan menghitung semua pernyataan kondisional, dengan rumus: V(G) = E – N + 2 atau V(G) = P + 1.

Dimana E adalah jumlah edge grafik alir, N adalah jumlah node grafik alir, dan P adalah jumlah node predikat yang ada dalam grafik alir G.

Menentukan sebuah basis set dari jalur independen linier. Nilai V(G) memberikan batas atas jumlah jalur independen linier melalui struktur program pengendalian.

Menyiapkan test case yang akan memaksa pelaksanaan setiap jalur di basis set. Data harus dipilih sehingga kondisi-kondisi di node predikat adalah set yang tepat saat masing-masing jalur diuji. Setiap set case dieksekusi dan dibandingkan dengan hasil yang diharapkan. Setelah semua test case telah selesai, penguji dapat yakin bahwa semua pernyataan dalam program ini telah dilaksanakan setidaknya sekali