**BAB II**

**TINJAUAN PUSTAKA**

* 1. **Gambaran Umum Pura Besakih**
     1. **Sejarah Pura Besakih**

Pura besakih terletak di Kecamaatan Rendang, Kabupaten Karangasem Provinsi Bali. Berdiri dikaki Gunung Agung. Pura Besakih adalah pura terbesar di Bali. Wisatawan asing sering menyebutnya dengan sebutan the mother temple of Bali. Pura besakih adalah komplek pura, dimana Pura Penataran Agung merupakan pusat dari pura – pura yang ada di dalam area pura Besakih tersebut.

Besakih asal katanya dari kata Basuki yang artinya selamat, kemudian lazim di sebut Besakih atau Basakih. Pura Besakih adalah huhusan 86 buah pura. Komplek Pura Besakih terdiri atas delapan belah buah pura umum, empat pura Catur Lawa, sebelas pura pedarman, enam pura non-pedarman, 29 pura dadia, tujuh pura berkaitan dengan pura dadia dan sebelas pura lainnya.

Pura Besakih asal mulanya didirikan oleh Rsi Markandya adalah seorang Yogi dari India yang tinggal di Jawa Timur tepatnya di Gunung Rawung. Karena ketinggian ilmu bhatinnya, kesucian rohaninya, serta kecakapan dan kebijaksanaan beliau maka oleh rakyat, beliau diberi julukan Bhatara Giri Rawang.

Beliau juga seorang pertapa. Mulanya beliau bertapa di Gunung Demulung, sekian waktu kemudian beliau bertapa ke Gunung Hyang (Dieng di Jawa Tengah). Sekian waktu lamanya bertapa, akhirnya beliau mendapat pawisik wahyu dari Tuhan agar merabas hutan du Pulau Dawa (Bali) untuk kemudian dibagi – bagikan kepada para pengikutnya.

Demikianlah kemudian beliau berangkat ke tanah Bali diserta pengikutnya yang pertama yang berjumlan 8000 orang dengan perlengkapan dan peralatan yang diperlukan. Sesampainya di tempat yang dituju, beliau memerintahkan pengikutnya agar mulai merambas hutan. Tempat itu merupakan hutan belantara yang di tumbuhi kayu yang lebat dan semak belukar. Pada saat itu Pulau Bali belum terpisah seperti sekarang dengan Pulau Jawa. Artinya selat bali (segara rupek) belum ada pada saat kedatangan Rsi Markandya ke

Sekian lama merambas hutan, karena pada saat mulai merambas hutan itu tidak didahului dengan upacara (yadnya) maka murkalah Sang Hyang Widhi. Para pengikut Rsi Markandya banyak yang sakit dan meninggal, juga tidak sedikit yang dimakan binatang buas. Sang Rsi sangat berduka. Kemudian beliau memerintahkan pengikutnya untuk menghentikan perambasan. Dengan hati yang sedih beliau kemudian mengajak pengikutnya untuk kembali ke Jawa. Beliau kembali ke tempat pertapanya semula untuk mohon petunjuk kepada sang Hyang Widhi.

Setelah beberapa lamanya beliau berada dipertapaannya, timbul cita – citanya kembali untuk melanjutkan merambas hutan tersebut. Pada suatu hari yang baik, beliau kembali berangkat ke tanah Bali. Kali ini beliau mengajak pengikutnya yang kedua berjumlah 4000 orang yang berasal dari desa Aga yaitu penduduk yang mediami lereng Gunung Rawung. Turut dalam rombongan beserta alat – alat pertanian dan bibit tanaman untuk di tanam di tempat yang baru.

Sesampainya di tempat yang di tuju, Rsi Markandya beserta para Pandita atau para Rsi melakukan yoga samadhi, weda samadhi, melakukan upacara Dewa Yadnya dan Bhuta Yadnya serta Pratiwi Stawa disertai doa penolak seluruh hama. Selesai melakuka upacara lalu beliau memerintahkan pengikutnya mulai merambas hutan, menebangi kayu – kayu melai dari sebelah selatan menuju ke utara.

Ketika di rasa sudah cukup luas, kemudia Rsi Markandya memerintahkan pengikutnya menghentikan untuk dipergunkan sebagai : swah, tgalan, dan pekarangan rumah.

Demikianlah pengikut Rsi Markandya yang berasal dari Desa Aga (penduduk lereng gunung rawung jawa timur) menetap di tempat itu sampai sekarang. Ditempat bekas dimulainya perambasan hutasn itu oleh sang Rsi atau Yogi Markandya menanam kendi (caratan) berisi air disertai lima jenis logam yaitu : emas, perak, tembaga, perunggu dan besi di sebut Panca Datu dan permata Mirahadi (mirah yang utama) dengan di sitertai sarana upakara selengkapnya dan diperciki Tirta Pengentas (air suci). Tempat menanam lima jenis logam itu di beri nama Basuki yang artinya selamat. Kenapa disebut demikian, karena pada kedatangan Rsi Markandya yang ke dua beserta 4000 pengikutnya selamat tidak menemuihambatan atau bencana seperti yang di alami pada saat kedatangan beliau yang pertama. Di tempat itu kemudia didirikan pelinggih. Lambat laun di tempat itu kemudia didirikan pura atau khayangan yang di beri nama Pura Basukian. Pura inilah cikal – bakal berdirinya pura – pura yang lain di komplek Pura Besakih. ada pendapat yang mengatakan bahwa pembangunan pura ditempat itu dimulai sejak 85 atau 163 Masehi. Pembangunan kemplek pura di pura Besakih sifatnya bertahap dan berkelanjutan disertai usaha pemugaran dan perbaikan yang dilakukan secara terus menerus dari masa kemasa.

* + 1. **Pedarman**

Pedarman adalah pura untuk memuja leluhur yang sudah suci yang disebut Dewa Pitara. Di Besakih ada banyak Pura Pedarman. Di antaranya ada Catur Luwa yang ada empat sisi Pura yang menyebar ke empat penjuru, Pura Ratu Pasek, dengan warna Putih selaku penyade atau penanggung jawab di Pura Gelap, Pura Pande, dengan warna Merah selaku penyade atau penanggung jawab di Pura Kiduling Kreteg, Pura Ratu Penyarikan, dengan warna Kuning selaku penyade atau penanggung jawab di Pura Kulkul, Pura Ratu Dukuh, dengan warna Hitam selaku penyade atau penanggung jawab di Pura Batumedag, dan di lengkapi oleh Bujangga Sakti, Dalem Sukawati, Dalem Bakas, Kaba – Kaba, Mengwi, Arya Sukahet, Sri Arya Kepakisan, Bujangga Waisnawa, Arya Kenceng, Arya Telabah Apit Yeh, Arya Kenuruhan. [2]

* 1. **Video Dokumenter**

Video dokumenter merupakan satu bentuk produk audio visual yang menceritakan suatu fenomena keseharian. Fenomena tersebut cukup pantas diangkat menjadi perenungan bagi penonton. Materi dokumenter dapat berupa cerita tentang keprihatinan sosial, pengalaman dan pergaulatan hidup yang memberikan inspirasi dan semangat hidup bagi penonton, atau kilas balik dan kupasan tentang peristiwa yang pernah terjadi dan ada kaitannya dengan masa sekarang. Video dokumenter tak pernah lepas dari tujuan penyebaran informasi pendidikan, dan propaganda bagi orang atau kelompok tertentu. Intinya video dokumenter tetap berpijak pada hal – hal senyata mungkin. Kunci utama dalam video dokumenter merupakan penyajian fakta. Video dokumenter terhubung dengan tokoh, peristiwa dan lokasi yang nyata. Video dokumenter merekam peristiwa yang sunguh – sungguh terjadi tidak menciptakan suatu kejadian. Dalam membuat video dokumenter terdapat kriteria dimana video tersebut bagus atau tidak [4]. Berikut ini kriteria video dokumenter yang bisa di katakan bagus :

1. Merupakan para pelaku yang sesungguhnya.
2. Tidak memiliki tokoh protagonis dan antagonis.
3. Struktur film sederhana.
4. Film berisi kenyataan atau fakta bukan rekayasa.
   1. **Bentuk Video Dokumenter**
5. Dokumenter Berdasarkan *Stock Shot*

Program dokumenter yang berdasarkan *stock shot* ini tinggal menyusun daftar *shot* yang diperlukan dan mencarinya di perpustakaan. Kekurangan *shot* tertentu dengan mudah diupayakan dengan pengambilan baru [4].

1. Dokumenter Yang Didramatisir

Format ini lebih sesuai menggunakan model *screenplay teatrikal* karena aspek visual dan aureal dapat diketahui sebelumnya dan dapat direncanakan seperti halnya sebuah drama yang disutradarai [4].

1. Dokumenter Model Instruksional

Jenis format ini termasuk dokumenter yang sebenarnya karena *shooting-*nya tidak dapat direncanakan cepat sebelumnya. Video dokumenter jenis ini banyak dirancang khusus untuk mengajari penonton bagaimana melakukan berbagai macam hal yang mereka ingin lakukan [4].

* 1. **Unsur – unsur Video Dokumenter**

1. Gambar (Visual)

Gambar yang diambil berdasarkan peristiwa tertentu. Orang – orang yang direkam dalam video tersebut, benar – benar ada dan pernah ada, bukan sebagai pemeran yang menggantikan seseorang dalam video [4].

1. Kata – kata (Verbal)

Kata – kata dalam video dokumenter berasal dari penuturan langsung dari subjek yang menjadi tokoh dalam video dokumenter tersebut. Kata – kata yang dilontarkan biasanya berupa kesaksian atas sejarah maupun peristiwa tertentu. Namun kata – kata tersebut juga bisa berasal dari narator atau narasumber untuk menggambarkan peristiwa maupun memberikan keterangan tertentu pada tempat – tempat yang di rekam dalam gambar [4].

* 1. **Tipe – Tipe Video Dokumenter**

1. Tipe *Expository*

Tipe ini yang paling klasik dibandingkan yang lain karena banyak digunakan untuk film dokumenter yang ditayangkan oleh televisi pada masa sekarang. Pada tahun 1930-an, tokoh besar dokumenter, *John Grierson* menawarkan sebuah bentuk yang sangat berbeda dari dokumenter yang sebelumnya yang dianggap terlalu puitik. Tawaran tersebut adalah paparan yang berupa penjelasan (*explanation*) yang bersamaan dengan gambar – gambar di film. Menurutnya, dengan menggunakan paparan yang menjelaskan maka pembuat film dokumenter bisa “naik kelas” dari yang semula mengangkat tema – tema masalah sosial di Dunia. Perbedaan yang tajam dengan film dokumenter yang dianggap puitik seperti yang di buat oleh *Joris Ivens* (*The Bridge* dan *Rain*) ataupun *Dziga Vertov* (*A Man With A Movie Camera*) adalah pada penggunaan aspek visual dan cara berceritanya. Dasar pemikiran dari perbedaan itu adalah penekanannya pada isi film yang cenderung *retorik* ataupun tujuannya yaitu penyebaran informasi secara *persuasive* [4].

1. Tipe *Observational*

Tipe *observational* merupakan tipe film dokumenter yang *filmmaker*-nya menolak untuk mengintervensi objek dan peristiwanya. Tipe ini juga menolak untuk menggunakan narasi (*voice-of-god*), komentar dari luar ruang cerita, wawancara, bahkan menolak penggunaan tulisan panjang yang menjelaskan adegan (*intertitles*) seperti yang digunakan *Robert Flaherty* dalam film *Nanook of the North*. Penekannanya untuk memeparkan potongan kehidupan manusia secara langsung.

1. Tipe *Interactive*

Tipe dokumenter *interactive* menjadi kebalikan dari dokumenter *observational*, dimana pada *observational*, *filmmaker* tidak pernah atau tidak boleh tampak di dalam filmnya. Sedangkan tipe *interactive*, *filmmaker*-nya menampakkan diri secara mencolok dilayar dan sering melibatkan diri pada peristiwa serta berinteraksi dengan subjeknya. Aspek utama dari dokumenter *interactive* adalah wawancara, terutama dengan subjek – subjeknya sehingga bisa didapatkan komentar – komentar dan respon langsung dari narasumbernya.

1. Tipe *Reflexive*

*Filmmaker* dalam dokumenter *reflexive* sudah melangkah satu tahap lebih maju dibandingkan tipe *interactive*. Tujuannya untuk membuka “kebenaran” lebih lebar kepada penontonnya. Tipe ini lebih memfokuskan pada bagaimana film itu dibuat. Artinya, penonton dibuat menjadi sadar akan adanya unsur – unsur film dan proses pembuatan film tersebut, justru hal inilah yang menjadi titik perhatiannya.

1. Tipe *Performative*

Tipe film dokumenter ini berciri paradoksal, dimana pada satu sisi tipe ini justru mengalihkan perhatian penonton dari “dunia” yang tercipta dalam film. Sedangkan sisi yanglain justru menarik perhatian penonton pada aspek ekspresi dari film itu sendiri. Tujuannya untuk mempresentasikan “dunia” dalam film secara tidak langsung. Juga menciptakan suasana dan nuansa “tradisi” dalam film yang cukup kental yaitu tradisi penciptaan subjek atau peristiwa dalam film fiksi. Aspek penciptaan tersebut bertujuan untuk menggambarkan subjek atau peristiwanya secara lebih subjektif, lebih ekspresif, lebih stylistik, lebih mendalam serta lebih kuat menampilkan penggambarannya. Subjek dan peristiwa tersebut dibuat secara baik dan terasa lebih hidup sehingga penonton dapat merasakan pengalaman dari peristiwa yang dibuat.

1. Tipe *Poetic*

Video atau film dokumenter tipe *poetic* cendrung memiliki interpretasi subjektif pada subjek – subjeknya. Pendekatan dari tipe ini mengabaikan kandungan penceritaan tradisional yang cenderung menggunakan karakter tunggal (*individual characters*) dan peristiwa yang harus di kembangkan. *Editing* dalam dokumenter *poetic* sangat nyata bahwa kesinambungan (*continuity*) tidak memiliki dampak apapun sebab dalam *editing*-nya lebih mengeksplorasi asosisasi dan pola yang melibatkan ritme dalam waktu (*temporal rhythms*) dan jukstaposisi ruang (*spatial juxtapositions*). Tipe *poetic* mengilustrasikan kesan subjektif tanpa kandungan argumentasi apapun. Hal ini sering dianggap sebagai salah tau gerakan garda epan (*avant-garde*) [4].

* 1. **Teknik Pengambilan Gambar**
     1. **Sudut Pengambilan Gambar**

1. *Bird Eye View Shoot*

Teknik *Bird Eye View Shoot* adalah teknik pengambilan gambar yang dilakukan dengan ketinggian kamera berada di atas ketinggian objek. Hasilnya akan terlihat lingkungan yang luas dan benda – benda lain tampak kecil dan berserakan [4].

1. *High Angel*

Sudut pengambilan gambar dari arah atas objek sehingga mengesankan objek jadi terlihat kecil. Teknik ini memiliki kesan dramatis yaitu nilai “kerdil”.

1. *Low Angel*

Sudut pengambilan gambar dari arah bawah objek sehingga mengesankan objek jadi terlihat besar. Teknik ini memiliki kesan dramatis yaitu nilai agung atau *prominance*, berwibawa, kuat dan dominan.

1. *Eye Level*

Sudut pengambilan gambar sejajar dengan objek. Hasilnya memperlihatkan tangkapan pandangan mata seseorang.

1. *Frog Level*

Sudut pengambilan gambar dengan ketinggian kamera sejajar dengan alas/dasar kedudukan objek atau lebih rendah. Hasilnya akan tampak seolah – olah mata penonton mewakili mata katak [4].

* + 1. **Ukuran Gambar**

1. *Extreme Close Up*

Sudut pengambilan gambar dengan ketinggian kamera sejajar dengan alas atau dasar kedudukan objek atau lebih rendah.

1. *Big Close Up*

Pengambilan gambar dari sebatas kepala hingga dagu.

1. *Close Up*

Gambar diambil dari jarak dekat, hanya sebagian dari objek yang terlihat seperti hanya mukanya saja atau sepasang kaki yang bersepatu baru.

1. *Medium Close Up*

Pengambilan dari jarak sedang, jika objeknya orang maka yang terlihat hanya separuh badannya saja (dari dada keatas).

1. *Mid Shoot*

Pengambilan dari jarak sedang, jika objeknya orang maka yang terlihat hanya separuh badannya saja (dari pinggang keatas).

1. *Kneet Shoot*

Pengambilan gambar objek dari kepala hingga lutut.

1. *Extreme Long Shoot*

Gambar diambil dari jarak sangat jauh, yang ditonjolkan bukan objek lagi tetapi latar belakangnya.

1. *Full Shoot*

Pengambilan gambar objek dari kepala hingga kaki.

1. *Long Shoot*

Pengambilan gambar secara keseluruhan. Gambar diambil dari jarak jauh, seluruh objek terkena hingga latar belakang objek.

1. *One Shoot*

Pengambilan gambar satu objek.

1. *Two Shoot*

Pengambilan gambar dua objek.

1. *Three Shoot*

Pengambilan gambar tiga objek.

1. *Group Shoot*

Pengambilan gambar sekelompok orang [4].

* 1. **Aplikasi**

Definisi aplikasi adalah penggunaan dalam suatu komputer, intruksi (*instruction*) atau pernyataan (statement) yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi output [5].

Aplikasi berasal dari kata application, yang memiliki makna penerapan/pengguna. Aplikasi merupakan salah satu jenis perangkat lunak komputer siap pakai yang memanfaatkan kemampuan komputer untuk menjalankan tugas yang diinginkan pengguna. Berbeda dengan perangkat lunak sistem, aplikasi menggabungkan beberapa kemampuan komputer, namun tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan tugas yang menguntungkan pengguna.

* 1. **Android**

Android adalah Sistem Operasi yang digunakan untuk perangkat berbasis mobile seperti handphone atau biasa disebut smartphone, komputer tablet, netbook, media player portable, dan lain-lain. Android dikembangkan dibawah LINUX dan Open Handset Alliance (OHA) yang berarti siapapun dapat mengembangkan android di hardware mereka. The Android Open Source Project (AOSP) yang dipimpin oleh Google, mempunyai tugas untuk perbaikan sampai dengan pengembangan ke depan dari android [6].

Android memiliki beberapa versi dan berikut adalah beberapa versi Android dengan menggunakan nama-nama kue dari keluaran pertama hingga versi terkini:

1. Android versi 1.1

Pada 9 Maret 2009, Google merilis Android versi 1.1. android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, voice search (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

1. Android versi 1.5 (Cupcake)

Pada pertengahan mei 2009, Google kembali merilis telpon seluler dengan menggunakan Android dan SDK (*Softwere Development KIt*) dengan versi 1.5 (*cupcake*) sebagai penyempurna dari versi sebelumnya. Pada Android *cupcake* terdapat penambahan berupa fasilitas menggugah *youtube*, aplikasi *headset* nirkabel *bluetooth*, tampilan *keyboard* dilayar serta tampilan gambar bergerak yang lebih aktraktif.

1. Android Versi 1.6 (Donut)

Pada bulan September 2009 diluncurkannya Android 1.6 (*Donut*) dengan dengan fitur yang lebih baik dari pendahulunya, yaitu menayangkan indicator baterai pada ponsel dan control applet VPN, dapat menentukan fle yang akan dihapus, *zoomin*-*zoomout* pada gambar, penggunaan koneksi CDMA dan lain-lain.

1. Android Versi 2.0/2.1 (Éclair)

Pada 3 Desember 2009 kembali diluncurkan ponsel Android versi 2.0/2.1 (Eclair). Versi ini mulai di pakai banyak *Smartphone*, fitur utama éclair adalah perubahan total struktur dan tampilan *userinterface* serta versi android pertama yang mendukung format HTML5. Diantaranya adalah peningkatan Google Map 3.1.2, perubahan UI pada *browser* baru, daftar kontak baru, dukungan flash pada kamera 3,2 MP,Digital zoom dan Bluetooth 2.1.

1. Android versi 2.2 (Froyo: Frozen Yoghurt)

Pada 20 mei 2010 Android versi 2.2 Froyo diluncurkan. Perubahan-perubahan terhadap versi-versi sebelumnya antara lain dukungan *AdobeFlash* 10.1, kecepatan aplikasi dan kinerja 2 sampai 5 kali lebih cepat, integrasi V8 *Javascriptengine* yang dipakai google *chrome* untuk mempercepat kemampuan *rendering* pada *browser*, pemasangan aplikasi dalam SD Card, kemampuan Wifi *HotspotPorTabel*, dan kemampuan *autoupdate* dalam aplikasi Android *Market*.

1. Android versi 2.3 (Gingerbread)

Pada 6 Desember 2010, Android kembali merilis versi terbaru yaitu Android 2.3 *Gingerbread*. Kelebihan versi ini sudah pasti lebih aktraktif dan sudah mampu mendukung fitur dual kamera untuk melakukan *video call* serta meningkatkan mutu aplikasi-aplikasi permainan berbasis *mobile* Android.

1. Android versi 3.0/3.1 (Honeycomb)

Pada bulan mei 2011 Android versi 3.0/3.1 *HoneyComb* dirilis. Android *Hoeycomb* merupakan sistem operasi Android yang tujuannya dikhususkan untuk penggunaan Tabelt berbasis Android.

1. Android versi 4.0 (ICS: Ice Cream Sandwich)

Android versi 4.0 ICS di rilis pada tanggal 19 oktober 2011 di Hongkong. Android ICS dapat digunakan pada *Smartphone* maupun Tabelt. Fitur utama Android ICS adalah *interface* dan ukuran layar standar (*nativeScreen*) beresolusi 720p (*HighDefinition*).

1. Android Versi 4.1 (Android Jelly Bean)

Android versi 4.1 merupakan Android versi yang cukup laris. Fitur terbaru dari Android *JellyBean* ini salah satunya adalah peningkatan kemampuan *on*-*screenkeyboard* yang lebih cepat serta lebih *responsive*.

1. Android Versi 4.4 (Android KitKat)

Android versi 4.4 ini diluncurkan pada September 2013 merupakan Android dengan fitur dan peningkatan *resolution* grafik.

1. Android Versi 5.0 (Android Lolipop)

Android versi 5.0 ini diluncurkan pada Juni 2014 merupakan Android terbaru dengan beberapa perbaharuan di *user interface* yang lebih minimalis dan beberapa tambahan seperti fitur *art runtime, volta* dan tambahan 15 bahasa baru.

* 1. **Adobe Flash Cs 6**

Program Adobe Flash CS6 menyediakan berbagai macam fitur yang akan sangat membantu para animator untuk membuat animasi menjadi semakin mudah dan menarik. Dengan fitur – fitur yang ada pada program Adobe Flash CS6, akan menjadikan program Adobe Flash sebagi program animasi dan presentasi yang makin digemari oleh para animator – animator di dunia. Program Adobe Flash Pro CS6 telah mampu mengolah teks maupun objek dengan efek tiga dimensi sehingga tampak lebih menarik [7].

* 1. **Action Script**

Sama dengan bahasa pemrograman yang lain, *ActionScript* berisi banyak elemen yang berbeda serta strukturnya sendiri. Kita harus merangkainya dengan benar agar *ActionSrcript* dapat menjalankan dokumen sesuai dengan keinginan. Jika tidak merangkai semuanya dengan benar, maka hasil yang didapatkan akan berbeda atau file flash tidak akan bekerja sama sekali. *ActionScript* juga dapat diterapkan untuk *action* pada *frame*, tombol, *movie clip*, dan lain-lain.

Salah satu fungsi *ActionScript* adalah memberikan sebuah konektivitas terhadap sebuah objek, yaitu dengan menuliskan perintah-perintah didalamnya [7].

* 1. **Adobe Premiere Pro**

Adobe Premiere Pro adalah sebuah program penyunting video. Adobe Premiere adalah bagian dari Adobe Creative Suite, sebuah rangkaian dari desain grafis, *video editing*, dan pengembangan aplikasi web yang dibuat oleh Adobe *System*. Premiere Pro mendukung banyak kartu video *editing* dan *plug-in* untuk mempercepat proses, tambahan mendukung format *file*, dan video atau audio efek.

Adobe Premiere Pro merupakan penerus untuk mendesain ulang premiere, dan diluncurkan pada tahun 2003. Premiere Pro merujuk ke versi dirilis pada tahun 2003 dan kemudian, sedangkan Premiere merujuk pada rilis sebelumnya. Meskipun dua versi Premiere Pro hanya didukung *Windows*, Premiere Pro CS3 tersedia baik untuk *Windows* dan *Mac OS* (hanya berbasis Intel *Mac* yang didukung), membuatnya menjadi salah satu dari beberapa intas *platform* *NLEs* tersedia [8].

* 1. **Adobe After Effect Cs 6**

Adobe After Effect adalah salah satu *software Composting* yang populer dan telah digunakan secara luas dalam pembuatan video, multimedia, film dan wed. After Effect terutama dipakai dalam penambahan efek khusus seperti efek petir, hujan, salju, ledakan bom, dan efek khusus lainnya. After Effect telah membantu para praktisi perfilman *Hollywood* dalam menghasilkan film-film dengan efek khusus yang spektakuler. *Spawn, Titanic, Deep Rising, The Jackal* adalah beberapa judul film yang telah memanfaatkan potensi After Effect. Adanya kesamaan *interface* antara After Effect dengan keluarga Adobe yang lain seperti PhotoShop memberikan kemudahan dan kenyamanan dalam pemakaiannya. Integrasi After Effect bersama dengan PhotoShop, Illustrator, dan Premiere akan menghasilkan karya yang sebelumnya sulit dibayangkan untuk diwujudkan. Pada versi terbarunya, After Effect 6. Berbagai fitur yang lebih memudahkan bagi para profesional dalam menghasilkan efek khusus yang inovatif namun dengan waktu yang terbatas. Adobe After Effect memiliki alat untuk membuat *shape* (seperti yang terdapat pada Adobe photoShop). Pada Adobe After Effect terdapat *keyframe* seperti yang terdapat pada Adobe Flash (cara menganimasikannya juga hampir sama). Terdapat juga *Expression* yang hampir mirip dengan *ActionScript* pada flash [8].

* 1. **Adobe Photoshop Cs 6**

Adobe Photoshop CS6 adalah salah satu *software* khusus yang digunakan untuk mengolah gambar, mengoreksi warna *image*, memperbaiki *image*, menggabungkan beberapa *image*, memberi efek khusus seperti sorotan cahaya, tetesan air, efek timbul pada *image*, disamping itu terdapat pula beberapa fasilitas *painting* untuk melukis.Kemudahan dan fleksibilitas kemampuan perangkat lunak ini tidak diragukan lagi dan banyak direkomendasikan sebagai pengolah grafis berkwalitas tinggi serta menjadi standar industri. *Software* Adobe Photoshop terbaru saat ini yang digunakan adalah Adobe Photoshop CS6.

* 1. **Corel Draw X6**

CorelDraw adalah perangkat lunak aplikasi yang berfungsi sebagai *editor* grafik berbasis *vektor*. Objek *vektor* seringkali dijadikan sebagai andalan oleh para praktisi dan profesional pada bidang desain grafis. Aplikasi CorelDraw digunakan untuk membuat *layout* bagi media cetak sampai dengan membuat gambar *vektor* yang halus dengan tingkat ketajaman gambar yang tinggi.

CorelDraw dibuat oleh perusahaan perangkat lunak bernama Corel yang berada di Ottawa, Kanada. Versi terbaru sampai tulisan ini dibuat adalah CorelDraw versi 16 atau lebih populer dengan CorelDraw X6. CorelDraw awalnya dikembangkan untuk dijalankan pada sistem operasi *Windows* 2000 keatas dan sempat juga dibuat versi CorelDraw untuk *Linux* dan *Mac OS*, akan tetapi proses pengembangannya dihentikan, dikarenakan faktor tingkat penjualan yang rendah. [9].

* 1. **Storyboard**

Secara umum, *storyboard* merpakan gambar sketsa yang dibuat pada panel – panel berbentuk segi empat, yang disusun berurutan dan saling berkelanjutan membentuk sebuah alur cerita sesuai dengan naskah. Isi *storyboard* merupakan penggabungan narasi (teks) dan visual (gambar) yang terkoordinasi satu sama lain. Dengan kata lain, *storyboard* merupakan naskah yang divisualkan melalui media gambar sketsa. Definisi tersebut menunjukkan bahwa *storyboard* berfungsi untuk memudahkan penyampaian suatu ide cerita kepada orang lain. Elemen gambar yang tersaji pada *storyboard* mampu menggugah imajinasi atau khayalan seseorag, untuk kemudian diarahkan menuju persepsi yang sama dengan ide cerita [10].

* 1. **Flowchart**

Flowchart merupakan kumpulan dari gambar yang biasanya berupa simbol atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya. Flowchart adalah kumpulan dari simbol gambar yang menunjukan arah atau flow dari proses terhadap data. Simbol-simbol dalam flowchart dibagi menjadi 2 kelompok :

1. **Flow Direction Symbols**

Dipakai untuk menggabungkan antara simbol yang satu dengan simbol lainnya.

Tabel 2.1 Flow Direction Symbols

|  |  |  |
| --- | --- | --- |
| No | Simbol | Penjelasan Simbol |
| 1 |  | ***Symbol Off-line Connector*** ( Simbol untuk keluar/masuk prosedure atau proses dalam lembar/halaman yang lain) |
| 2 |  | ***Symbol Connector*** (Simbol untuk keluar/masuk prosedur atau proses dalam lembar/halaman yang sama)  ***Symbol Comunication Link*** (Simbol transmisi untuk informasi dari satu lokasi ke lokasi lainnya) |

1. **Processing Symbols**

Menunjukkan jenis operasi pengolahan dalam suatu prosedur

Tabel 2.2 Processing Symbols

|  |  |  |
| --- | --- | --- |
| No | Simbol | Penjelasan Simbol |
| 1 |  | ***Symbol Process***(Simbolyang menunjukkan pengolahan yang dilakukan oleh komputer) |
| 2 |  | ***Symbol Manual Operation*** (Simbol yang menunjukkan pengolahan yang tidak dilakukanoleh komputer) |
| 3 |  | ***Symbol Decision*** (Simbol untuk kondisi yang akan menghasilkan beberapa kemungkinan jawaban/aksi) |
| 4 |  | ***Symbol Predefined Process*** (Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage) |
| 5 |  | ***Symbol Terminal*** (Simbol untuk permulaan atau akhir dari suatu program) |
| 6 |  | ***Symbol Off-line Storage*** (Simbol yang menunjukkan bahwa data di dalam symbol ini akan disimpan) |
| 7 |  | ***Symbol Manual Input***  (Simbol untuk pemasukan data secara manual on-line keyboard) |
| 8 |  | ***Symbol Keying Operation*** (Simbol operasi dengan menggunakan mesin yang mempunyai keyboard) |

* 1. **UML (Unified Modeling Language)**

Unified Modeling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek [11].

* + 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, 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. *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 meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya.

Tabel 2.3 Simbol Use Case Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | *Actor* | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*. |
| 2 |  | Dependency | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*Independent*) akan mempengaruhi elemen yang bergantung padanya atau elemen yang tidak mandiri (*Dependent*). |
| 3 |  | Generalization | Hubungan dimana obyek anak (*descendent*) berbagi perilaku dan struktur data dari obyek yang ada di atasnya atau obyek induk (*ancestor*). |
| 4 | <<uses>> | Include | Menspesifikasikan bahwa use case sumber secara eksplisit. |
| 5 | <<extends>> | Extend | Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan. |
| 6 |  | Association | Apa yang menghubungkan obyek satu dengan obyek yang lainnya. |
| 7 | Systemm | System | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 8 |  | Use Case | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor. |
| 9 |  | Collaboration | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |
| 10 |  | Note | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi |

*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 keluarfungsionalitas 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.

* + 1. **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.

*Class* memiliki tiga area pokok :

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

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
3. *Public*, dapat dipanggil oleh siapa saja

Tabel 2.4 Simbol Class Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | *Activity* | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain. |
| 2 |  | *Action* | State dari sistem yang mencerminkan eksekusi dari suatu aksi. |
| 3 |  | *Initial Node* | Bagaimana objek dibentuk atau diawali. |
| 4 |  | *Activity Final Node* | Bagaimana objek dibentuk dan diluncurkan. |
| 5 |  | *Fork Node* | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran. |
| 6 |  | *Decision* | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. |
| 7 |  | *Swimlane* | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi. |

* + 1. **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. *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.

Tabel 2.5 Simbol Activity Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Gambar** | **Nama** | **Keterangan** |
| 1 | ; Object1 | Objek | Objek merupakan instance dari sebuah *class* dan tertulis serta tersusun secara horizontal. Digambarkan oleh sebuah class dengan nama objek didalamnya yang diawali dengan sebuah titik koma (;). |
| 2 | Actor1 | *Actor* | Actor juga dapat berkomunikasi dengan objek, maka actor juga dapat diurutkan sebagai kolom. Simbol actor sama dengan simbol actor pada *use case diagram*. |
| 3 |  | Lifeline | Lifeline mengidentifikasikan keberadaan sebuah objek dalam baris waktu. Notasi untuk Lifeline adalah garis putus-putus vertical yang ditarik dari sebuah objek. |
| 4 |  | *Activation* | Dinotasikan sebagai sebuah kotak segiempat yang digambar pada sebuah lifeline. Activation mengidentifikasi sebuah objek yang akan melakukan sebuah aksi. |
| 5 | Message 1 | *Message* | Digambarkan dengan anak panah horizontal antara activation. Message mengidentifikasikan komunikasi antara objek-objek. |

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.

* + 1. **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 scenario 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.

Tabel 2.6 Simbol Sequence Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | Generalization | Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya atau objek induk (ancestor). |
| 2 |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari dua objek. |
| 3 | |  | | --- | | Class | | - Attribute 1 | | +Methode1() | | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |
| 4 |  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |
| 5 |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |
| 6 |  | Dependency | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*Independent*) akan mempengaruhi elemen yang bergantung padanya atau elemen yang tidak mandiri (*Dependent*). |
| 7 |  | *Association* | Apa yang menghubungkan obyek satu dengan obyek yang lainnya. |

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.*

### 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, 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. *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 meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya.

Tabel 2.3 Simbol Use Case Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | *Actor* | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*. |
| 2 |  | Dependency | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*Independent*) akan mempengaruhi elemen yang bergantung padanya atau elemen yang tidak mandiri (*Dependent*). |
| 3 |  | Generalization | Hubungan dimana obyek anak (*descendent*) berbagi perilaku dan struktur data dari obyek yang ada di atasnya atau obyek induk (*ancestor*). |
| 4 | <<uses>> | Include | Menspesifikasikan bahwa use case sumber secara eksplisit. |
| 5 | <<extends>> | Extend | Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan. |
| 6 |  | Association | Apa yang menghubungkan obyek satu dengan obyek yang lainnya. |
| 7 | Systemm | System | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 8 |  | Use Case | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor. |
| 9 |  | Collaboration | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |
| 10 |  | Note | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi |

*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 keluarfungsionalitas 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.

### 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.

*Class* memiliki tiga area pokok :

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

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
3. *Public*, dapat dipanggil oleh siapa saja

Tabel 2.4 Simbol Class Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | *Activity* | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain. |
| 2 |  | Action | State dari sistem yang mencerminkan eksekusi dari suatu aksi. |
| 3 |  | Initial Node | Bagaimana objek dibentuk atau diawali. |
| 4 |  | Activity Final Node | Bagaimana objek dibentuk dan diluncurkan. |
| 5 |  | Fork Node | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran. |
| 6 |  | Decision | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. |
| 7 |  | Swimlane | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi. |

### Activity Diagram

*Activity diagrams* 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. *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.

Tabel 2.5 Simbol Activity Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Gambar** | **Nama** | **Keterangan** |
| 1 | ; Object1 | Objek | Objek merupakan instance dari sebuah *class* dan tertulis serta tersusun secara horizontal. Digambarkan oleh sebuah class dengan nama objek didalamnya yang diawali dengan sebuah titik koma (;). |
| 2 | Actor1 | *Actor* | Actor juga dapat berkomunikasi dengan objek, maka actor juga dapat diurutkan sebagai kolom. Simbol actor sama dengan simbol actor pada *use case diagram*. |
| 3 |  | Lifeline | Lifeline mengidentifikasikan keberadaan sebuah objek dalam baris waktu. Notasi untuk Lifeline adalah garis putus-putus vertical yang ditarik dari sebuah objek. |
| 4 |  | *Activation* | Dinotasikan sebagai sebuah kotak segiempat yang digambar pada sebuah lifeline. Activation mengidentifikasi sebuah objek yang akan melakukan sebuah aksi. |
| 5 | Message 1 | *Message* | Digambarkan dengan anak panah horizontal antara activation. Message mengidentifikasikan komunikasi antara objek-objek. |

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.

### 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 scenario 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.

Tabel 2.6 Simbol Sequence Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | Generalization | Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya atau objek induk (ancestor). |
| 2 |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari dua objek. |
| 3 | |  | | --- | | Class | | - Attribute 1 | | +Methode1() | | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |
| 4 |  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |
| 5 |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |
| 6 |  | Dependency | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*Independent*) akan mempengaruhi elemen yang bergantung padanya atau elemen yang tidak mandiri (*Dependent*). |
| 7 |  | *Association* | Apa yang menghubungkan obyek satu dengan obyek yang lainnya. |

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.*

* 1. **Black-Box Testing**

*Black box* testing adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian kotak hitam dilakukan dengan membuat kasus uji yang 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 [12].

*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 performasi, kesalahan inisialisasi dan akhir program.