**BAB II**

**TINJAUAN PUSTAKA**

**2.1 Pengertian Media Informasi**

Media informasi dapat disimpulkan sebagai alat untuk mengumpulkan dan menyusun kembali sebuah informasi sehingga menjadi bahan yang bermanfaat bagi penerima informasi, adapun penjelasan Sobur (2006) media informasi adalah alat-alat grafis, fotografis atau elektronik untuk menangkap, memproses, dan menyusun kembali informasi.

**2.2 Kabupaten Banyuwangi**

Kabupaten Banyuwangi merupakan kabupaten yang terletak di ujung paling Timur pulau Jawa dan berbatasan langsung dengan Kabupaten Situbondo di utara, selat Bali di Timur, Samudra Hindia di selatan serta Kabupaten Jember dan Kabupaten Bondowoso di barat. Kabupaten Banyuwangi dihuni oleh beragam suku bangsa. Mayoritas penduduk lokal Banyuwangi adalah suku Osing yang dipercaya merupakan sub-suku Jawa, dan suku lain yang hidup dengan damai seperti, suku Madura, suku Jawa, Bali dan Bugis.

Dalam keseharian, penduduk lokal memakai bahasa Osing, yang merupakan ragam tertua bahasa Jawa tapi berdasarkan kebudayaan, budaya suku Osing banyak dipengaruhi oleh budaya Bali. Kabupaten Banyuwangi selain perlintasan dari Jawa ke Bali, juga merupakan daerah pertemuan berbagai jenis kebudayaan daru berbagai wilayah. Budaya masyarakat Banyuwangi diwarnai oleh budaya Jawa, Bali dan Madura. Kekayaan budaya yang ada sudah sepatutnya untuk dijaga dan dilestarikan[1].

**2.2.1 Suku Osing**

Suku Osing merupakan penduduk asli banyuwangi karena suku Osing adalah masyarakat yang hidup pada pemerintahan kerajaan Blambangan. Suku Osing juga memiliki adat istiadat budaya, bahasa yang berbeda dari masyarakat Jawa dan Madura. Suku Osing atau lebih dikenal dengan Wong Osing memiliki bahasa sendiri yakni bahasa Osing yang merupakan turunan langsung dari bahasa jawa kuno tapi bukan merupakan bahasa jawa karena dialegnya yang berbeda[3].

**2.2.2 Tradisi Kabupaten Banyuwangi**

Kabupaten Banyuwangi memiliki beragam tradisi yang sampai saat ini masih berjalan dan di jaga dengan baik. Diantaranya adalah :

1. Tradisi Kebo-keboan

tradisi selamatan desa sebagai ungkapan rasa syukur atas hasil panen yang melimpah sekaligus sebagai upacara bersih desa agar seluruh warga diberi keselamatan dan dijauhkan dari segala marabahaya. Gambar tradisi Kebo-Keboan dapat dilihat pada tabel 2.1.

1. Tradisi Petik Laut

Tradisi petik laut adalah sebuah upacara adat atau ritual sebagai rasa syukur kepada Tuhan, dan untuk memohon berkah rezeki dan keselamatan yang dilakukan oleh para nelayan pesisir pantai desa Muncar. Gambar tradisi Petik Laut dapat dilihat pada tabel 2.1.

1. Tumpeng Sewu merupakan ritual adat selamatan massal yang telah berlangsung turun temurun pada suku Osing di Desa Kemiren, sebagai ungkapan rasa syukur atas rezeki yang diberikan Tuhan Yang Maha Esa yang mereka terima selama satu tahun. Gambar tradisi Tumpeng Sewu dapat dilihat pada tabel 2.1.
2. Tradisi Rebo Wekasan

Tradisi Rebo Wekasan adalah sebuah tradisi memperingati hari Rabu terakhir di bulan Safar. Tujuan peringatan itu adalah menolak bencana, talak balak. Kegiatan yang dilakukan berkisar pada berdoa, Shalat Sunnah, bersedekah. Gambar tradisi dapat dilihat pada tabel 2.1.

1. Tradisi Barong Ider Bumi

barong ider bumi merupakan ritual pengusir bahaya dan permohonan kesuburan kepada Sang Maha Kuasa. Selain itu, upacara ini dipercaya sebagai ritual pensucian diri dari segala kesalahan yang dilakukan selama setahun dan penyembuhan terhadap wabah penyakit[4]. Gambar tradisi Barong Ider Bumi dapat dilihat pada tabel 2.1.

* + 1. **Tari Kabupaten Banyuwangi**

Kabupaten Banyuwangi selain memiliki berbagai macam tradisi juga memiliki bermacam-macam kesenian tari. Diantara adalah :

1. Tari Gandrung

Tari Gandrung Banyuwangi adalah seni tari yang berasal dari kabupaten banyuwangi tari Gandrung ini dibawakan sebagai perwujudan rasa syukur masyarakat setiap habis panen. Gambar Tari Gandrung dapat dilihat pada tabel 2.1.

1. Tari Seblang

Tari Seblang merupakan keragaman adat yang dilakukan dari generasi ke generasi. Tari Seblang digunkan untuk ritual bersih desa dengan tujuan seupaya desa selalu dalam kondisi aman dan tentram. Gambar Tari Seblang dapat dilihat pada tabel 2.1.

1. Tari Barong Banyuwangi

Tari Barong merupakan bentuk seni pertunjukan rakyat yang menggunakan madia barong. Kesenian ini diyakini sangat sakral, sehingga ada perlakuan khusus. Gambar Tari Barong Banyuwangi dapat dilihat pada tabel 2.1.

1. Tari jaranan Buto

Tari Jaranan Buto adalah Tari ini menggunakan properti kuda buatan yang berwajah raksasa atau Buto begitu pula dengan para pemainnya yang juga menggunakan tata rias muka layaknya seorang raksasa yang lengkap dengan muka merah bermata besar, bertaring tajam, berambut panjang dan gimbal. Gambar Tari Jaranan Buto dapat dilihat pada tabel 2.1.

1. Tari Puput Banyu

Tari Puputan Banyu Merupakan Tari Banyuwangi bernuansa perang. Puput Bayu merupakan perang terkejam di Banyuwangi antara para penduduk asli Blambangan dan VOC Belanda. [4]. Gambar Tari Puputan Banyu dapat dilihat pada tabel 2.1.

Tabel 2.1 Tradisi dan Tari

|  |  |
| --- | --- |
| C:\Users\Tollo\Pictures\BANYUWANGI\kebo-keboan by gobanyuwangi.jpg  Gambar 2.1 Tradisi Kebo-keboan (Sumber :[5]) | C:\Users\Tollo\Pictures\BANYUWANGI\Petik-Laut-Muncar-jawa by beritadaerah.jpg  Gambar 2.2 Tradisi Petik Laut (Sumber :[5]) |
| C:\Users\Tollo\Pictures\BANYUWANGI\oooo.jpg  Gambar 2.3 Tradisi Tumpeng Sewu (Sumber :[5]) | C:\Users\Tollo\Pictures\BANYUWANGI\rebo wekasan.jpg  Gambar 2.4 Tradisi Rebo Wekasan (Sumber :[5]) |
| C:\Users\Tollo\Pictures\BANYUWANGI\oooi.jpg  Gambar 2.5 Tradisi Barong Ider Bumi (Sumber :[6]) | C:\Users\Tollo\Pictures\BANYUWANGI\chef-img.png  Gambar 2.6 Tari Gandrung (Sumber :[7]) |
| C:\Users\Tollo\Pictures\BANYUWANGI\seblang.jpg  Gambar 2.7 Tari Seblang (Sumber :[8]) | C:\Users\Tollo\Pictures\BANYUWANGI\ooooii.jpg  Gambar 2.8 Tari Barong Banyuwangi (Sumber :[9]) |
| C:\Users\Tollo\Pictures\BANYUWANGI\jaranan buto yakso birowo.jpg  Gambar 2.9 Tari Jaranan Buto (Sumber :[10]) | C:\Users\Tollo\Pictures\BANYUWANGI\tari puputan banyu.jpg  Gambar 2.10 Tari Perang Puputan Banyu (Sumber :[11]) |

**2.3 Android versi 4.2 Jelly Bean**

Android Jelly Bean diluncurkan pada 10 Mei 2011. Android versi ini meningkatkan *input keyboard*, desain baru fitur pencarian, UI yang baru dan pencarian melalui *Voice Search* yang lebih cepat. Versi ini juga dilengkapi Google Now. Fitur photo *sphere* untuk panaroma, *daydream* sebagai *screensaver, power control, lock screen widget*, widget terbaru. Android 4.2 Pertama kali dikenalkan melalui LG Google Nexus 4[12].

**2.4 Android AVD**

Android Virtual Device (AVD) adalah konfigurasi dari emulator sehingga kita dapat menjalankan perangkat Android sesuai model yang dipilih, misalnya Android versi 1.5 atau versi 2.2. Setiap AVD terdiri dari :

1. Sebuah profil perangkat keras, Kita dapat mengatur opsi untuk menentukan fitur *hardware* emulator. Misalnya, kita dapat menentukan apakah menggunakan perangkat kamera, apakah menggunakaan *keyboard* *qwerty* fisik atau tidak, berapa banyak memori internal, dan lain-lain.
2. Sebuah pemetaan versi Android. Kita dapat menentukan versi dari platform android akan berjalan pada emulator.
3. Pilihan lainnya, Kita dapat menentukan skin yang digunakan pada emulator, yang memungkinkan dimensi layar, tampilan, dan sebagainya. Kita juga dapat menentukan SDCar virtual untuk digunakan dengan emulator [13].

**2.5 Eclipse**

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*). Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, [Python](http://id.wikipedia.org/wiki/Python), [Perl](http://id.wikipedia.org/wiki/Perl), [PHP](http://id.wikipedia.org/wiki/PHP), dan lain sebagainya. Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya. Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open*source, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan plug-in[13].

**2.6 Android SDK (Software Development Kit)**

*Software Development Kit* (SDK) Android meliputi seperangkat alat pengembangan, ini termasuk *debugger*, *library*, sebuah *handset emulator*, dokumentasi, contoh kode, dan *tutorial*. Saat ini *platform* pengembangan yang didukung termasuk computer yang menjalankanLinux, Mac OS X 10.4.0 atau yang terbaru, Windows XP atau yang terbaru. Lingkungan pengembangan terpadu (IDE) adalah Eclipse yang sudah dilengkapi dengan Android *Development Tools* (ADT) Plug-in, meskipun pengembangan dapat menggunakan editor teks lain untuk mengedit file XML Java. Aplikasi Android yang dikemas dalam format APK dan disimpan di folder aplikasi pada Android OS (folder hanya dapat diakses dalam mode super user / root untuk alasan keamanan)[13].

**2.7 Web Service**

Web service adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan. Web service digunakan sebagai suatu fasilitas yang disediakan oleh suatu web site untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan-layanan (service) yang disediakan oleh suatu sistem yang menyediakan web service. Web service menyimpan data informasi dalam format XML, sehingga data ini dapat diakses oleh sistem lain walaupun berbeda platform, sistem operasi, maupun bahasa compiler[14].

**2.7.1 Arsitektur Web Service**

Web service 3 Entitas dalam Arsitekturnya, yaitu:

1. Service Provider: Berfungsi untuk menyediakan layanan/service dan mengolah sebuah registry agar layanan-layanan tersebut dapat tersedia.
2. Service Registry: Berfungsi sebagai lokasi central yang mendeskripsikan semua layanan/service yang telah di register.
3. Service Requestor: Peminta layanan yang mencari dan menemukan layanan yang dibutuhkan serta menggunakan layanan tersebut[14].

**2.7.2 CSS (*Cascading Style Sheet)***

*Cascading Style Sheet* (CSS) merupakan aturan untuk mengendalikan beberapa komponen dalam sebuah web sehingga akan lebih terstruktur dan seragam. CSS bukan merupakan bahasa pemograman. Sama halnya *style* dalam [aplikasi](http://id.wikipedia.org/wiki/Aplikasi) pengolahan kata seperti [Microsoft Word](http://id.wikipedia.org/wiki/Microsoft_Word) yang dapat mengatur beberapa *style*, misalnya *heading,* *subbab, bodytext, footer, images,* dan *style* lainnya untuk dapat digunakan bersama-sama dalam beberapa berkas (file). Pada umumnya CSS dipakai untuk memformat tampilan halaman web yang dibuat dengan [bahasa](http://id.wikipedia.org/wiki/Bahasa) [HTML](http://id.wikipedia.org/wiki/HTML) dan [XHTML](http://id.wikipedia.org/wiki/XHTML). CSS dapat mengendalikan [ukuran](http://id.wikipedia.org/wiki/Ukuran) [gambar](http://id.wikipedia.org/wiki/Gambar), [warna](http://id.wikipedia.org/wiki/Warna) bagian tubuh pada teks, warna [tabel](http://id.wikipedia.org/wiki/Tabel), ukuran border, warna border, warna [hyperlink](http://id.wikipedia.org/wiki/Hyperlink), warna [mouse](http://id.wikipedia.org/wiki/Mouse) over, spasi antar paragraf, spasi antar teks, margin kiri, kanan, atas, bawah, dan parameter lainnya. CSS adalah bahasa *style sheet* yang digunakan untuk mengatur tampilan dokumen. Dengan adanya CSS memungkinkan kita untuk menampilkan halaman yang sama dengan format yang berbeda[14].

**2.7.3 PHP (*Hypertext Preprocessor)***

PHP diciptakan pertama kali oleh Rasmus Lerdorf pada tahun 1994. Awalnya PHP digunakan untuk mencatat jumlah serta untuk mengetahui siapa saja pengunjung pada *home* pagenya. Rasmus Lerdorf adalah salah satu seorang pendukung open source. Oleh karena itu, ia mengeluarkan Personal Home Page Tools versi 1.0 secra gratis kemudian menambah kemampuan PHP 1.0 dan meluncurkan PHP 2.0. Pada tahun 1996, PHP telah banyak digunakan dalam *website* di dunia. Sebuah kelompok pengembang *software* yang terdiri dari Rasmus, Zeew Suraski, Andi Gutman, Stig Bakken, Shane Caraveo, dan Jim Winstead bekerja sama untuk menyempurnakan PHP 2.0. Akhirnya, pada tahun 1998, PHP 3.0 diluncurkan. Penyempurnaan terus dilakukan sehingga pada tahun 2000 dikeluarkan PHP 4.0.

PHP memiliki banyak kelebihan yang tidak dimiliki oleh bahasa *script* sejenis. PHP difokuskan pada pembuatan *script server side*, yang bisa melakukan apa saja yang dapat dilakukan oleh CGI, seperti mengumpulkan data dari form, menghasilkan isi halaman web dinamis, dan kemampuan mengirim serta menerima cookies, bahkan melebihi kemampuan CGI. PHP dapat digunakan pada semua sistem operasi, antar lain Linux, UNIX (termasuk variannya HP-UX, Solaris, dan OpenBSD),Microsoft Windows, Mac OS X, RISC OS. PHP juga banyak mendukung banyak webserver, seperti Apache, *Microsoft Internet Information Server* (MIIS), Personal *Web Server* (PWS), Netscape and iPlanet Servers, Oreilly Website Pro Server, Audium, Xitami, OmnilHTTPd, dan masih banyak lagi yang lainnya, bahkan PHP dapat bekerja sebagai suatu CGI processor.

PHP tidak terbatas pada hasil keluaran HTML (*Hypertext Markup Languages*). PHP juga memiliki kemampuan untuk mengolah keluaran gambar, file PDF, dan movies flash. PHP juga dapat menghasilkan text seperti XHTML dan file XML lainnya [14].

**2.8 MySQL**

MySQL adalah sebuah perangkat lunak database (basis data) sistem terbuka yang sangat terkenal dikalangan pengembang sistem database dunia yang di gunakan untuk berbagai aplikasi terutama untuk aplikasi berbasis web. MySQL mempunyai fungsi sebagai SQL (*Structured Query Language*) yang di miliki sendiri dan telah di perluas. MySQL umumnya digunakan bersamaan dengan PHP untuk membuat aplikasi yang dinamis dan *powerful* [14].

**2.9 Database SQLite**

SQLite merupakan sebuah system manajemen basis data relasional yang bersifat ACID-compliant dan memiliki ukuran pustaka kode yang *relative* kecil. Ditulis dalam bahasa C. SQLite bersifat public domain yang dikerjakan oleh D.Richard Hipp.

SQLite bukanlah sebuah sistem yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integrasi dari sebuah program secara keseluruhan sehingga *protocol* komunikasi utama yang digunakan adalah melalui pemanggilan API secara langsung melalui bahasa pemrograman. Seluruh elemen basisdata (definisi data, *table, indeks*, dan data) disimpan sebagai sebuah *file* [14].

**2.10 Analisa Dan Desain Sistem**

Analisa system bertujuan untuk menganalisa system yang sedang Berjalan, sehingga dapat dipahami bagaimana kondisi system telah ada. Pada tahap desain system menggunakan DFD, ERD, Class Diagram, Sequence Diagram [15].

**2.10.1 Data flow diagram**

Data Flow Diagram (DFD) memberikan gambaran alur bagaimana data masuk dan keluar dari dalam dan ke suatu entity/representasi dari sumber dan tujuan aliran data tersebut, aturan dari proses data, penyimpanan data dan *entitas eksternal*. Selain itu DFD merupakan diagram yang menggambarkan sistem secara terstruktur dengan membagi–bagi menjadi beberapa level.

Pada pembuatan DFD juga harus memperhatikan tahap-tahapnya. Adapun tahapan di dalam pembuatan DFD adalah sebagai berikut:

1. Diagram Konteks (*level Top*)

Diagram yang menggambarkan mengenai sistem secara global. Dalam hal ini ditetapkan entitas-entitas eksternal yang terlibat dalam proses, baik sebagai sumber maupun tujuan. Diagram konteks menggambarkan sistem dalam satu lingkaran dan hubungan dengan entitas luar­­­. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem.

1. Diagram Level 0

Diagram yang memberikan gambaran mengenai proses-proses apa saja yang akan dilakukan dan melibatkan entitas-entitas eksternal yang ada serta *data store* tertentu. Diagram konteks juga menggambarkan tahapan-tahapan proses yang ada pada diagram konteks.

1. Diagram Level Detail (Level 1)

Diagram yang digunakan untuk menggambarkan arus data secara detail dari tahapan-tahapan proses pada diagram nol. Dalam menggambarkan diagram detail ini perlu diperhatikan konsistensi jumlah input dan output yang ada pada diagram nol dan detail. Bila pada level ini sudah tidak ada lagi terdapat anak proses, maka pada proses dinamakan proses *primitive*, pada nomor prosesnya cukup diberikan P [15]. Simbol-simbol DFD dapat dilihat pada Tabel 2.2.

Tabel 2.2 Simbol Notasi DFD (Sumber :[15])

|  |  |  |
| --- | --- | --- |
| **Simbol** | **Nama Simbol** | **Penjelasan** |
|  | *Data Flow*  (alur data) | Menunjukkan alur data informasi /objek) yang mengalir. Alur data digunakan untuk menerangkan perpindahan data/paket data dari satu bagian ke bagian lainnya. |
|  | *Process*  (proses) | Menunjukkan tugas atau proses yang dilakukan baik secara manual atau otomatis. Komponen proses yang menggambarkan transformasi input menjadi output. Penamaan proses disesuaikan dengan proses/kegiatan yang sedang dilakukan. |
|  | *External Entity*  (terminator/  entitas eksternal) | Menunjukkan tempat asal data (sumber) atau tempat tujuan data (tujuan). Nama terminator/entitas eksternal ditulis dalam bentuk tunggal. |
| ­­­­ | *Data Store* (penyimpanan data) | Simbol ini menunjukkan gudang data. Komponen ini digunakan untuk membuat model sekumpulan paket data. |

**2.10.2 Entity Relationship Diagram (ERD)**

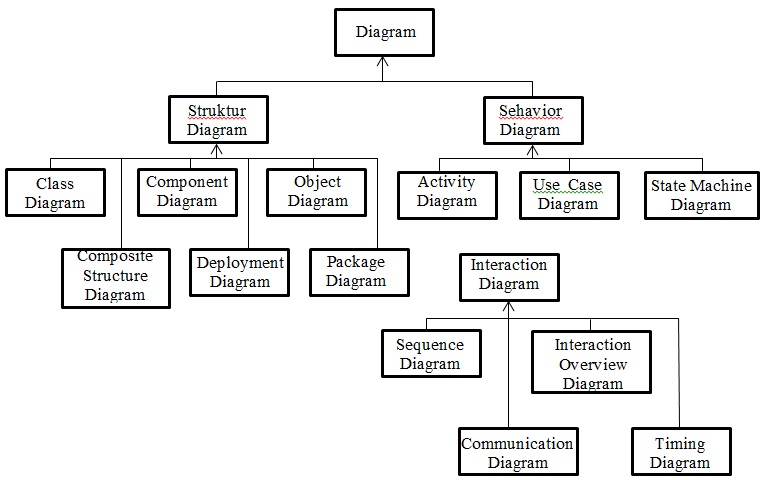
Model entity–relationship berisi komponen–komponen dari suatu himpunan entitas dan relasi yang masing–masing dilengkapi dengan atribut–atribut yang merepresentasikan seluruh fakta yang ditinjau sehingga dapat diketahui hubungan antara *entity–entity* yang ada dengan atribut–atributnya. Selain itu juga bisa menggambarkan hubungan yang ada dalam pengolahan data, seperti hubungan *many to many, one to many* atau *one to one* [15]. Simbol-simbol ERD dapat dilihat pada Tabel 2.3.

Tabel 2.3 Simbol Notasi ERD (Sumber :[15])

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **ARTI** | **KETERANGAN** |
|  | *Entity* (entitas) | Objek yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain. |
|  | *Relationship* (relasi) | Hubungan yang terdiri antara satu entitas atau lebih dan tidak dapat berdiri sendiri. |
|  | *Attribute* (atribut) | Elemen data yang menunjukkan ciri entitas atau karakter dari entitas. |
|  | *One to one* (satu ke satu) | Setiap entitas pada himpunan entitas A dapat berelasi dengan paling banyak satu entitas pada himpunan entitas B, demikan juga sebalikanya. |
|  | *One to many* (satu ke banyak) | Bentuk relasi dari entitas A yang berjumlah satu dengan entitas B yang berjumlah banyak. |
|  | *Many to one* (banyak ke satu) | Hubungan entitas A yang berjumlah banyak dengan entitas B yang berjumlah satu. |

**2.10.3 Pengertian UML (Unified Modelling Language)**

UML adalah kependekan dari Unified Modelling Language yang merupakan suatu cara untuk menyelesaikan suatu masalah dengan mendeskripsikannya yang telah mejadi standar dalam dunia industri untuk memvisualisasikan, merancang dan mendokumentasikan system perangkat lunak. Dengan menggunakan UML kita dapat membangun model untuk segala bentuk dan jenis aplikasi perangkat lunak, yang mana aplikasi dapat berjalan pada perangkat lunak dengan system operasi dan jaringan apapun. Gambar struktur UML dapat dilhat pada gambar 2.11.



Gambar 2.11 Struktur UML (Sumber :[15])

Dalam suatu proses pengembangan software, analisa dan rancangan telah merupakan teknologi yang sangat tua. Pada saat masalah ditelusuri dan spesifikasi dinegosiasikan, dapat dikatakan kita berada pada tahap rancangan. Merancang adalah menemukan suatu cara untuk menyelesaikan masalah, salah satu tool / model untuk merancang pengembangan *software* yang berada *object oriented* adalah UML.[15]

1. Bagian-Bagian UML

Bagian-bagian utama dari UML, berikut ini pengertian dari masing-masing bagian-bagian utama dari UML yaitu :

1. **View**

Digunakan untuk melihat sistem yang dimodelkan dengan aspek yang berbeda. View bukan untuk melihat grafik melainkan suatu abstraksi yang berisi diagram.

1. Use case View

Merupakan fungsionalitas sistem yang seharusnya dilakukan sesuai dengan keinginan pengguna atau user.

1. Logical View

Merupakan fungsionalitas dari sistem yang terjadi ketika object mengirimkan pesan ke object lainnya dengan fungsi tertentu.

1. Concurrency view

Merupakan sistem dalam proses yang digambarkan dalam diagram dinamis dan diagram implementasi yang digunakan untuk pengembangan, pengintegrasi, dan penguji.

1. **Diagram**

Berbentuk diagram grafik yrng menunjukan sebuah symbol elemen yang disusun untuk mengilustrasikan suatu bagian dari sistem. Diagram merupakan bagian dari view yang ketika digambarkan biasanya dialokasikan unutk view tertentu[15].

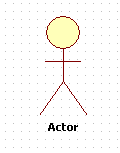
1. Diagram-Diagram pada UML

Berikut adalah jenis-jenis diagram pada UML yaitu :

a. Use Case Diagram

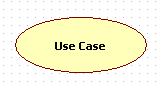
Use case adalah abstraksi dari interaksi antara sistem dan *actor*. Use case bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Use case merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata user. Sedangkan use case diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan client.

1. Actor Pada dasarnya *actor* bukanlah bagian dari *use case diagram*, namun untuk dapat terciptanya suatu *use case diagram* diperlukan beberapa *actor. Actor* tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Gambar *Actor* dapat dilihat pada gambar 2.12.



Gambar 2.12 Lambang Actor

2. Case gambaran fungsionalitas dari suatu sistem, sehingga customer atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun. Gambar Case dapat dilihat pada gambar 2.13.



Gambar 2.13 Lambang Usecase

3. Extend Relasi yang digunakan jika use case yang satu mirip dengan use case yang lain. Kelakuan yang hanya berjalan di bawah kondisi ertentu seperti menggerakan alarm. Gambar Extend dapat dilihat pada gambar 2.14.



Gambar 2.14 Lambang Extend

4. Include Relasi jika terdapat perilaku yang digunakan pada beberapa usecase. Kelakuan yang harus terpenuhi agar sebuah event dapat terjadi, dimana pada kondisi ini sebuah usecase adalah bagian dari usecase lainnya. Gambar Include dapat dilihat pada gambar 2.15.



Gambar 2.15 Lambang Include

b. 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 (metode/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 yaitu : Nama (stereotype), Atribut, Metode.

Masing-masing kelas harus mempunyai nama yang unik. Sebagian besar organisasi mempunyai konvensi penamaan sendiri untuk menamakan kelas-kelas yang dibuatnya. Umumnya kelas-kelas dinamakan menggunakan kata benda tunggal.

Nama kelas tidak menggunkan spasi. Ini dilakukan karena alasan praktis, dimana beberapa bahasa pemrograman tidak membolehkan adanya spasi. Hal lainnya yang perlu diperhatikan adalah bahwa nama kelas hendaknya pendek, cukup untuk menjelaskan apa yang akan kelas lakukan.

Jadi penamaan kelas sangat tergantung pada organisasi kita. Jika kita mempunyai kelas yang digunakan dalam organisasi yang bersangkutan, tetapi yang jelas bahwa hal tersebut harus konsisten digunakan untuk keseluruhan kelas-kelas yang dibuatnya.

Atribut dan Metode dapat memiliki salah satu sifat dan disini akan dijelaskan beberapa sifatnya tersebut, antara lain adalah :

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.

*Class* merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansisasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*. Simbol nitotasi Class Diagram dapat dilihat pada Tabel 2.4[15]

Tabel 2.4 Simbol Notasi *Class Diagram* (Sumber :[15])

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1 |  | *Generalization* | Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
| 2 |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek. |
| 3 |  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |
| 4 |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
| 5 |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |
| 6 |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri |
| 7 |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya |

1. Sequence Diagram

*Sequence Diagram* merupakan gambaran secara grafis bagaimana berbagai objek berinteraksi satu sama lain melalui pesan (*messages*) saat melaksanakan sebuah operasi atau sebuah *usecase*.

*Object* digambarkan dengan garis vertikal putus-putus yang disebut dengan *lifeline*. Simbol notasi Sequence Diagram dapat dilihat pada Tabel 2.5[15].

Tabel 2.5 Simbol Notasi *Sequence Diagram* (Sumber :[15])

|  |  |  |
| --- | --- | --- |
| **NAMA** | **KETERANGAN** | **GAMBAR** |
| *Object* | *Object* merupakan *instance* dari sebuah *class* dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah *class* dengan nama objek di dalamnya yang diawali dengan sebuah titik koma. |  |
| *Actor* | *Actor* juga dapat berkomunikasi dengan objek, maka *actor* juga dapat diurutkan sebagai kolom. Simbol *actor* sama dengan simbol pada *actor use case diagram*. |  |
| *Lifeline* | *Lifeline merupakan* mengindikasikan keberadaan sebuah objek dalam basis waktu. Notasi untuk *lifeline* adalah garis putus-putus vertical yang ditarik dari sebuah objek. |  |
| *Activation* | *Activation* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. *Activation* mengindikasikan sebuah objek yang akan melakukan sebuah aksi. |  |
| *Message* | *Message* digambarkan dengan anak panah horizontal antara *Activation*. *Message* sebagaimengindikasikan komunikasi. |  |

1. Activity Diagram

Pada dasarnya diagram *Activity* sering digunakan oleh *flowchart.* Diagram ini berhubungan dengan diagram *Statechart*. Diagram Statechart berfokus pada obyek yang dalam suatu proses (atau proses menjadi suatu obyek), diagram Activity berfokus pada aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain. Simbol yang ada pada activity diagram dapat dilihat pada tabel 2.6[15].

Tabel 2.6 Simbol Notasi Activity Diagram (Sumber :[16])

