BAB II
TINJUAN PUSTAKA

2.1 Sistem Informasi

Sistem informasi adalah sistem dapat didefinisikan dengan mengumpulkan, memproses, menyimpan, menganalisis, menyebarkan informasi untuk tujuan tertentu. Sistem informasi adalah sekumpulan subsistem yang saling berhubungan, berkumpul bersama-sama dan membentuk satu kesatuan, saling berintegrasi dan bekerjasama antara bagian satu dengan yang lainnya dengan cara-cara tertentu untuk melakukan fungsi pengolahan data, menerima masukan *(input)* berupa data-data, kemudian mengolahnya *(processing)*, dan menghasilkan keluaran *(output)* berupa informasi sebagai dasar pengambilan keputusan yang berguna dan mempunyai nilai nyata yang dapat dirasakan akibatnya baik pada saat itu juga maupun disaat mendatang, mendukung kegiatan operasional, manajerial, dan strategis organisasi, dan memanfaatkan berbagai sumber daya yang ada dan tersedia bagi fungsi tersebut guna mencapai tujuan [12]. Sistem Informasi adalah perangkat prosedur yang terorganisasi dengan sistematik, bila dilaksanakan akan menyediakan informasi yang dapat dimanfaatkan dalam proses pembuatan keputusan.[13]

Dari uraian beberapa para ahli informasi dapat disimpulkan sistem informasi adalah berupa proses pengolahan data yang menghailkan berupa informasi yang berfungsi untuk mencapai tujuan.

2.2 Sistem Informasi Gegrafis

Sistem Informasi Georafis atau *Georaphic Information Sistem* (GIS) merupakan suatu sistem informasi yang berbasis komputer, dirancang untuk bekerja dengan menggunakan data yang memiliki informasi spasial (bereferensi keruangan). Sistem ini mengcapture, mengecek, mengintegrasikan, memanipulasi, menganalisa, dan menampilkan data yang secara spasial mereferensikan kepada kondisi bumi. Teknologi SIG mengintegrasikan operasi-operasi umum database, seperti query dan analisa statistik, dengan kemampuan visualisasi dan analisa yang unik yang dimiliki

oleh pemetaan. Kemampuan inilah yang membedakan SIG dengan Sistem Informasi lainya yang membuatnya menjadi berguna berbagai kalangan untuk menjelaskan kejadian, merencanakan strategi, dan memprediksi apa yang terjadi. Berikut adalah beberapa definisi SIG yang telah beredar di berbagai sumber pustaka [14] :

1. Sistem Informasi Geografis (SIG) yakni sistem yang tersusun dari *software, hardware,* dan orang-orang yang mengelola, menyimpan, menampilkan, dan menganalisis data spesifik lokasi atau geografi. Selain itu, Sistem Informasi Geografis dirancang untuk memperoleh, menyimpan, memanipulasi, menganalis, mengelola, serta menyajikan data geografis. SIG dibedakan dari sistem informasi lain karena mengolah data dalam bentuk data spasial yang berorientasi geografis dan lokasi dengan koordinat tertentu, sehingga memungkinkannya untuk menjawab beberapa pertanyaan, termasuk lokasi dan informasi tentang tempat tersebut [15].
2. Sistem informasi geografis didefinisikan sebagai suatu alat atau media untuk memasukan, menyimpan, mengambil, memanipulasi, menganalisa dan menampilkan data-data beratribut geografis yang berguna untuk mendukung proses pengambilan keputusan dalam perencaan dan manajeman sumber daya alam, lingkungan, transportasi, masalah perkotaan dan administratif [16].
3. Sistem Informasi Geografis adalah kumpulan yang terorganisir dari perangkat keras berupa computer, perangkat lunak, metode, data geografis dan personil yang dirancang secara efisien untuk memperoleh, menyimpan, memperbaharui, menganalisis, memanipulasi, dan menampilkan semua bentuk yang bereferensi geografis [17].
4. Sistem Informasi Geografis (SIG) adalah suatu sistem yang dibuat berdasarkan pemetaan geografis bumi. Sistem ini dapat memberikan informasi mengenai letak dari lokasi tempat-tempat yang ada di permukaan bumi, serta memberikan keterangan-keterangan dari lokasi yang telah diberikan dan dapat memberikan informasi mengenai tempat lokasi wisata [18].

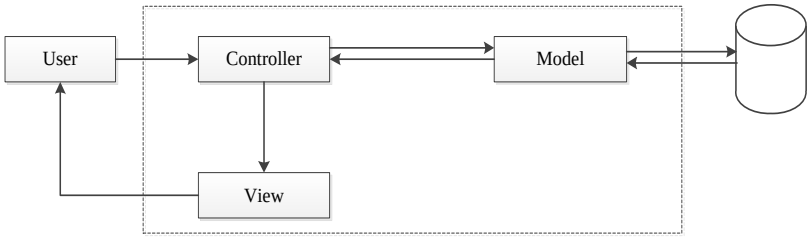
5. SIG yaitu sebagai sistem informasi yang digunakan untuk memasukan, menyimpan, memanggil kembali, mengolah, menganalisis dan menghasilkan data bereferensi geografis atau data geospatial, untuk mendukung pengambilan keputusan dalam perencanaan dan pengelolaan penggunaan lahan, sumber daya alam, lingkungan, tarnsportasi, fasilitas kota, dan pelayanan umum lainnya [19].

Dari beberapa definisi SIG di atas maka dapat disimpulkan bahwa SIG merupakan sebuah sistem atau teknologi berbasis komputer yang dibangun dengan tujuan untuk mengumpulkan, menyimpan, mengolah dan menganalisa, serta menyajikan data dan informasi dari suatu objek atau fenomena yang berkaitan dengan letak atau keberadaanya di permukaan bumi.

2.3 *Code Igniter*

CodeIgniter adalah *Framework* untuk bahasa pemprograman PHP, yang dibuat Rick Ellis pada tahun 2006. *CodeIgniter* memiliki banyak fitur yang membantu para pengembang PHP untuk dapat membuat aplikasi secara mudah dan cepat serta memiliki sifat yang fleksibel dapat mengembangkan dalam perangkat *web*, dekstop maupun *mobile"* [20].

CodeIgniter memilki konsep atau pola *Model-View-Controller* (MVC) sehingga kode-kode dapat di sederhanakan.



Gambar 2.1 Arsitektur MVC

2.3.1 *Web Based*

Web Based adalah aplikasi yang dibuat berbasis *web* yang membutuhkan *web server* dan *browser* untuk menjalankannya [21].

Dengan membuat sistem berbasis *web based* ada beberapa hal yang penting dan harus kita pikirkan sebelum membangun sistem tersebut, [22] diantaranya :

1. Tidak membutuhkan *hardware* dengan spesifikasi yang tangguh untuk menjalankan aplikasinya.
2. Server yang dibutuhkan cukup diinstallkan *tools* pendukung saja agar klien mudah menjalankan aplikasi
3. Infrastruktur jaringan yang dibutuhkan juga cukup besar karena aplikasi yang dibuat dapat diakses dari jaringan luar (internet).
4. Aplikasi berbasis *web based* dapat diakses dari berbagai perangkat dengan syarat menggunakan *web browser* saja sudah dapat mengaksesnya.
5. Jika aplikasi yang sudah jadi ingin di *update*, sangat mudah untuk melakukannya karena tidak membutuhkan membuka keseluruhan aplikasi.

2.3.2 PHP

PHP adalah salah satu bahasa script yang dieksekusi di sisi server web (server-side) yang didesain khusus untuk aplikasi web seperti halnya JSP, Perl (.pl), dan ASP. Script PHP dieksekusi di server dan menghasilkan output (jika ada) dalam bentuk HTML yang dikirimkan oleh server web ke client/browser (TI3).

PHP merupakan script untuk pemrograman script web server-side, script yang membuat dokumen HTML secara on the fly. Dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan editor teks atau editor HTML . Dengan menggunakan PHP maka maintenance suatu situs web menjadi lebih mudah. Proses update data dapat dilakukan

dengan menggunakan aplikasi yang dibuat dengan menggunakan script PHP [23].

Sehingga PHP merupakan bahsa pemprograman yang digunakan oleh pengembang untuk membuat sistem *website* dengan kumpulan bahasa HTML dan *script* lainya.

2.3.3 *MySql*

MySQL adalah singkatan dari *Structue Query Language* yang digunakan untuk mendefinisikan structure data, memodifikasi data pada basis data, menspesifikasi batasan keamanan (*security*), hingga pemeliharaan data . *Mysql* adalah RDBMS yang cepat dan mudah digunakan, serta sudah banyak digunakan untuk berbagai kebutuhan [24].

MySQL merupakan bahasa standar yang paling banyak digunakan untuk mengakses *database* relasional dan merupakan aplikasi yang dapat dipergunakan secara bebas.

2.4 Metode Pengembang Sistem

Metode pengembang sistem merupakan metode yang digunakan sebagai alur proses dalam pengembangan, sehingga penelitian dapat di kembangkan sesuai tahapan dari metode pengembang sistem.

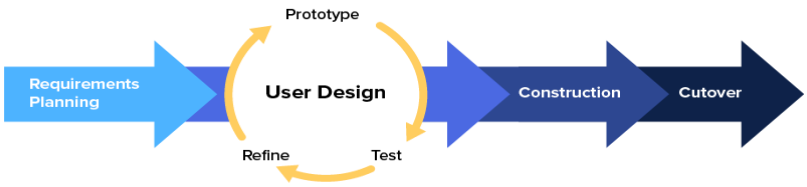
2.4.1 *Rapid Application Development* (RAD)

Menurut Pressman, RAD merupakan model proses lunak yang menekankan pada daur pengembangan hidup yang singkat. Sedangkan menurut Schach, Suatu model rapid *prototype*, merupakan *working* model dimana sebagian fungsional aplikasi sudah berjalan [25]. *Rapid Application Development* (RAD) adalah strategi siklus hidup yang ditujukan untuk menyediakan pengembangan yang jauh lebih cepat dan mendapatkan hasil dengan kualitas yang lebih baik dibandingkan dengan hasil yang dicapai melalui siklus tradisiona [26]. Dengan banyaknya *demand*, orang-orang di dunia IT harus mencari solusi untuk memenuhi permintaan tersebut. Metode ini merupakan semacam cikal bakal *agile project management*, karena bisa mengikuti *pace* bisnis yang

terus berkembang dan juga kebutuhan pasar yang terus meningkat. Pengembangan software pada umumnya seperti waterfall model membutuhkan perencanaan yang terbilang cukup kaku. Klien atau pelanggan seakan ‘dipaksa’ untuk menyetujui banyak hal di awal, tetapi mereka tidak bisa melihat proses pembuatannya.

Keuntungan utama menjalankan *rapid application development* adalah jangka waktu pengembangan lebih cepat. Hal ini dikarenakan *feedback* dari pelanggan cepat didapatkan dan semua perubahan yang dilakukan akan sesuai hasil tersebut. [27]

Akan tetapi, salah satu kekurangan RAD adalah kamu membutuhkan tim berisikan *developer* yang benar-benar memiliki *skill* tinggi dan juga metode ini hanya bisa digunakan untuk proyek yang bisa termodulasi.



Gambar 2.2 *Rapid Application Development* (RAD)

1. Kelebihan Model RAD

   Kelebihan metodologi RAD :

   a. Penghematan waktu dalam keseluruhan fase projek dapat dicapai.
   b. RAD mengurangi seluruh kebutuhan yang berkaitan dengan biaya projek dan sumberdaya manusia.
   c. RAD sangat membantu pengembangan aplikasi yang berfokus pada waktu penyelesaian projek.
   d. Perubahan desain sistem dapat lebih berpengaruh dengan cepat dibandingkan dengan pendekatan SDLC tradisional.
   e. Sudut pandang user disajikan dalam sistem akhir baik melalui fungsi-fungsi sistem atau antarmuka pengguna.
   f. RAD menciptakan rasa kepemilikan yang kuat di antara seluruh pemangku kebijakan projek.

2. Kelemahan Model RAD

Kelemahan pada pengembangan tersebut dapat dilihat berdasarkan kesesuaian pengembangan yang dilakukan, berikut adalah kelemahan metode pengembang sistem RAD :

a. Dengan metode RAD, penganalisis berusaha mepercepat projek dengan terburu-buru.
b. Kelemahan yang berkaitan dengan waktu dan perhatian terhadap detail. Aplikasi dapat diselesaikan secara lebih cepat, tetapi tidak mampu mengarahkan penekanan terhadap permasalahan-permasalahan perusahaan yang seharusnya diarahkan.
c. RAD menyulitkan *programmer* yang tidak berpengalaman menggunakan prangkat ini di mana *programmer* dan *analyst* dituntut untuk menguasai kemampuan-kemampuan baru sementara pada saat yang sama mereka harus bekerja mengembangkan sistem

2.4.2 Tahapan Penelitian

Tahapan dalam penelitian sebagai langkah-langkah penelitian yang harus dikerjakan, berikut adalah tahapan penelitian *Rapid Application Development.*

1. Tahap *Requirements Project*

   RAD dimulai dengan menentukan kebutuhan sebuah proyek (*project requirements*). Pada tahap ini, tim perlu menentukan kebutuhan yang ingin dipenuhi dari sebuah proyek. Kebutuhan ini tidak perlu spesifik. Tapi, sifatnya benar-benar umum dan jumlahnya bisa banyak. Baru dari situ, tim akan menentukan mana kebutuhan yang perlu diprioritaskan. Setelah mendapatkan kebutuhan yang jelas, barulah tim menentukan hal-hal yang lebih detail. Misalkan seperti tujuan, timeline, dan budget yang diperlukan.

2. Tahap Membuat *Prototype*

   Hal yang selanjutnya dilakukan adalah membuat *prototype. Developer* secepat mungkin akan membuat *prototype* dari aplikasi yang diinginkan. Lengkap dengan fitur dan fungsi yang berbeda-

beda. Tujuannya, sekadar untuk mengecek apakah *prototype* yang dibuat sudah sesuai dengan kebutuhan klien. Meski begitu, tahap ini bisa saja dilakukan berulang-ulang. Kadang juga melibatkan *user* untuk *testing* dan memberikan *feedback.* Proses ini memungkinkan tim mempelajari error yang mungkin muncul ke depannya. Ini berguna untuk mengurangi error dan *debugging*. Lewat tahapan ini, tim *developer* memiliki modal untuk membuat aplikasi yang mudah dipakai, stabil, tidak sering error, dan desainnya pun oke.

3. Proses Pengembangan dan Penngumpulan *Feedback*

   Setelah tahu aplikasi seperti apa yang ingin dibuat, *developer* mengubah *prototype* ke bentuk aplikasi versi beta sampai dengan final. Jadi, bisa dibilang tahap RAD inilah yang cukup intens. *Developer* terus-menerus melakukan *coding* aplikasi, melakukan testing sistem, dan integrasi dengan bagian-bagian lainnya. Karena itulah, *developer* menggunakan *tools* dan *framework* yang mendukung RAD agar cepat. Apalagi proses ini terus diulang sambil terus mempertimbangkan feedback dari klien. Baik itu soal fitur, fungsi, *interface*, sampai keseluruhan aspek dari produk yang dibuat. Nah, kalau prosesnya berjalan lancar, *developer* akan melanjutkan ke langkah berikutnya dengan finalisasi produk atau implementasi. Kalau pun tidak, proses ini kemungkinan akan terus diulang. Pun, kalau apes-apesnya aplikasi tidak tidak menjawab kebutuhan, *developer* akan kembali ke proses *prototyping*.

4. Tahap *Implementation* (implementasi).

   merupakan tahap pengujian terhadap aplikasi yang dikembangkan. Tahap ini programmer mengembangkan desain menjadi suatu program kemudian dilakukan proses pengujian untuk memeriksa kesalahan sebelum diaplikasikan.

2.5 Alat Pengembang Sistem

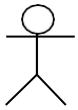
Unified Modelling Language adalah bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-

teks pendukung. Berikut ini merupakan penjelasan tentang masing-masing diagram yang ada pada *Unified Modelling Language* [28].

2.5.1 *Use Case Diagram*

Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut [28]. Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Use Case Diagram* dapat dilihat pada Tabel 2.1 :

Tabel 2.1 Simbol-simbol *Use Case Diagram*

No	**Simbol**	**Keterangan Fungsi**
1.	Aktor 	Aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.
2.	*Use Case* 	*Use Case* adalah deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
3.	Asosiasi	Asosiasi adalah apa yang menghubungkan antara objek satu dengan objek yang lainnya.
4.	Generalisasi 	Generalisasi adalah hubungan dimana objek anak (*descendent*) berbagi prilaku dan struktur data dari objek yang ada diatasnya atau sebaliknya dari bawah ke atas.
5.	*Defendency*	*Defendency* (ketergantungan) adalah hubungan dimana perubahan yang terjadi pada suatu elemen *defenden* (mandiri) akan mempengaruhi elemen yang bergantung padanya *(independen).*

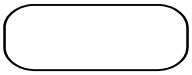
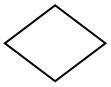
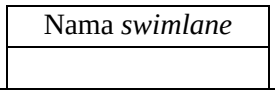
Sumber: [28]

2.5.2 *Activity Diagram*

Activity Diagram menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis atau menggambarkan aktivitas sistem bukan apa yang dilakukan aktor [28]. Berikut simbol-simbol yang akan

digunakan dalam menggambarkan *activity diagram* dapat dilihat pada Tabel 2.2:

Tabel 2.2 Simbol *Activity Diagram*

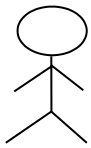
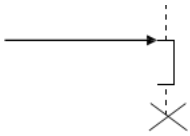
No.	Simbol	Keterangan
1.		Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.		Percabangan *(Decision)* merupakan asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.		Penggabungan *(Join)* merupakan asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.		*Swimlane* Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas.
6.		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

Sumber: [28]

2.5.3 *Squence Diagram*

Diagram squence menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dengan *massage* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup dalam diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. Berikut adalah simbol-simbol yang ada pada *squence* diagram pada Tabel 2.3.

Tabel 2.3 Simbol-simbol *Squence Diagram*

No.	Simbol	Deskripsi
1	Aktor  Atau Nama aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari akor adalah gambar orang, tapi akor belum tentu merupakan orang, biasanya dinyatakan dalam menggunakan kata benda di awal frase nama aktor
2.	Garis hidup /*lifeline* 	Menyataan kehidupan suatu objek
3.	Objek Nama objek: nama kelas	Menyetakan objek yang berinteraksi peran
4	Pesan tipe *destroy* <<*destroy*>> 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaliknya jika ada *create* maka ada D*estroy*

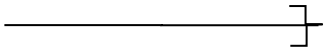
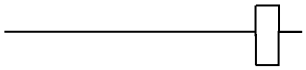
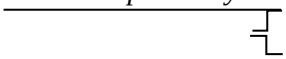
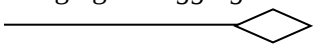
Sumber: [28]

2.5.4 *Class Diagram*

Class Diagram mengembangkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Class Diagram* dapat dilihat pada Tabel 2.4:

Tabel 2.4 Simbol *Class Diagram*

No.	Simbol	Deskripsi
1.	Nama_kelas *+Atribute* +Operasi	Kelas pada struktur sistem**.**
2.	Antar Muka/*Interface*  Nama_*Interface*	Sama dengan konsep interface dalam pemrograman berorientasi objek.

3.	Asosiasi / *Asociation*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan symbol
4.	Asosiasi Berarah / *Directed Association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan symbol.
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
6.	Ketergantungan / *dependency* 	Relasi antar kelas dengan makna ketergantungan antar kelas.
7.	Agregasi / *aggregation* 	Relasi antar kelas dengan maksna semua bagian *(whole-part)*

Sumber: [28]

2.6 Metode Pengujian Sistem

Metode pengujian sistem merupakan metode yang digunakan untuk melakukan testing pada sistem yang dibangun sehingga di peroleh hasil berupa sistem yang sesuai fungsinya.

2.6.1 Metode Pengujian *Black Box*

Black Box merupakan pengujian yang dapat dilakukan dengan melakukan pengamatan, pada hasil eksekusi melalui beberapa data uji dan memeriksa fungsional yang terdapat pada perangkat lunak. Jadi dapat kita dianalogikan seperti halnya kita melihat ke dalam kotak hitam, sehingga kita hanya bisa melihat tampilan luarnya saja tanpa kita tau apa yang ada didalam kotak hitam tersebut.

Sehingga sama seperti halnya dengan metode *Black Box* ini yang hanya dapat mengevaluasi dari tampilan luarnya dan fungsionalitasnya. Tanpa harus mengetahui apa sesungguhnya yang terjadi dalam proses detilnya. Pada pengetahuan khusus dari struktur kode internal dan

pengetahuan pada pemrograman dasar pada umumnya tidak diperlukan untuk *Black Box Testing*. Uji pada kasus yang dibangun disekitar spesifikasi dan persyaratan, yakni pada aplikasi yang seharusnya dilakukan.

2.6.2 Tahapan Pengujian Sistem

Tahapan pengujian sistem digunakan untuk mengetahui proses pengujian yang akan dilakukan, berikut adalah tahapan pengujian *Black Box*:

1. *Descision Table*

   Decision table adalah representasi tabel yang digunakan untuk mendeskripsikan dan menganalisis situasi keputusan prosedural di mana keadaan sejumlah kondisi menentukan pelaksanaan serangkaian tindakan.

2. *All-Pairs Testing*

   All-Pairs Testing atau disebut *pairwise testing* merupakan metode pengujian perangkat lunak kombinatorial yang digunakan untuk setiap pasangan parameter yang masuk kedalam sistem atau algoritma yang ada pada perangkat lunak.

3. *State Transition Table*

   State Transition Testing merupakan salah satu teknik dari black box testing. Teknik ini dilakukan dengan membuat test case yang menguji inputan yang sudah dibagi pada beberapa kelompok sesuai dengan fungsinya. Pengujian pada teknik ini dilakukan dengan berurutan sesuai dengan transisi, keadaan dan juga kejadian diantara inputan[29].

4. *Equivalence Partitioning*

   Equicalence Partitioning merupakan teknik yang membagi data masukan dari beberapa unit perangkat lunak menjadi beberapa partisi data dari mana *test case* dapat diturunkan. Pada prinsipnya, uji kasus ini dirancang untuk menutupi setiap partisi minimal.

Teknik ini digunakan untuk mendefinisikan kasus pengujian yang mengungkap kelas kesalahan, sehingga mengurangi jumlah pengujian yang harus dilakukan.

5. *Boundry Values Analysis*

   Boundary Value Analysis merupakan Pengujian yang dirancang untuk mencakup perwakilan dari batas Nilai-nilai batas. Pada nilai-nilai di sebuah partisi kesetaraan atau sebesar nilai terkecil di kedua sisi tepi.