**BAB II**

**LANDASAN TEORI**

* 1. **Aplikasi**

Menurut Henry (2004,12) ,  Aplikasi adalah suatu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktifitas seperti perniagaan, pelayanan masyarakat, periklanan atau semua proses yang dilakukan manusia. Menurut Supriyanto (2005,2), Aplikasi adalah program yang memiliki aktifitas pemrosesan perintah yang diperlukan untuk melaksanakan permintaan pengguna dengan tujuan tertentu.

Jadi dapat disimpulkan bahwa aplikasi adalah suatu perngkat lunak dalam suatu komputer atau yang digunakan untuk melayani kebutuhan akan beberapa aktifitas yang dapat mempermudah pekerjaan manusia dalam dengan tujuan tertentu.

* 1. **Badan Penyelengara Jaminan Produk Halal (BPJPH)**

BPJPH adalah badan yang dibentuk oleh Pemerintah untuk menyelenggarakan Jaminan Produk Halal (JPH) terhadap sejumlah produk yang dikonsumsi masyarakat. Penyelenggaraan tersebut berdasarkan amanat Undang – Undang Jaminan Produk Halal ( UU JPH )No 33 tahun 2014 yang bertujuan memberikan kenyamanan, keamanan, keselamatan, dan kepastian ketersediaan produk halal bagi masyarakat dalam mengonsumsi dan menggunakan produk dan meningkatkan nilai tambah bagi pelaku usaha untuk memproduksi dan menjual produk halal. Produk yang masuk, beredar, dan diperdagangkan di wilayah

Indonesia wajib bersertifikat halal. Sertifikat halal tersebut merupakan pengakuan kehalalan suatu Produk yang dikeluarkan oleh BPJPH berdasarkan fatwa halal tertulis yang dikeluarkan oleh Majelis Ulama Indonesia (MUI) dan kemudian menurut pasal 6 UU-JPH kewenangan BPJPH dalam menyelenggarakan jaminan produk halal mecakup :

1. Merumuskan dan menetapkan kebijakan JPH.

2. Menetapkan norma, standar, prosedur, dan kriteria JPH.

3. Menerbitkan dan mencabut Sertifikat Halal dan Label Halal pada Produk.

4. Melakukan registrasi Sertifikat Halal pada Produk luar negeri.

5. Melakukan sosialisasi, edukasi, dan publikasi Produk Halal.

6. Melakukan akreditasi terhadap LPH

7. Melakukan registrasi Auditor Halal.

8. Melakukan pengawasan terhadap JPH.

9. Melakukan pembinaan Auditor Halal.

10. Melakukan kerja sama dengan lembaga dalam dan luar negeri di bidang penyelenggaraan JPH.

Berdasarkan kewenangan atributif di atas, saat ini BPJPH merupakan satu-satunya lembaga yang berwenang untuk menyelenggarakan sertifikasi produk halal di Indonesia. Dengan demikian maka proses sertifikasi produk halal tidak lagi berada di bawah kewenangan Lembaga Pengkajian Pangan, Obat-obatan, dan Kosmetika Majellis Ulama Indonesia (LPPOM MUI). Meski demikian, sesuai dengan UU-JPH, MUI tetap mempunyai peran strategis dalam proses sertifikasi produk halal. Hal ini menunjukkan bahwa BPJPH dapat melakukan kerjasama dengan MUI terkait proses sertifikasi auditor halal, akreditasi Lembaga Penjamin Halal (LPH) dan penetapan kehalalan produk dalam bentuk fatwa(business-law.binus.ac.id).

* 1. **Sertifikasi Halal**

Sertifikat Halal adalah fatwa tertulis Majelis Ulama Indonesia yang menyatakan kehalalan suatu produk sesuai dengan syari’at Islam. Sertifikat Halal  MUI ini merupakan syarat untuk mendapatkan ijin pencantuman label halal pada kemasan produk dari instansi pemerintah yang berwenang.(Kemenag:2003).

Sertifikasi Halal MUI pada produk pangan, obat-obat, kosmetika dan produk lainnya dilakukan untuk memberikan kepastian status kehalalan, sehingga dapat menenteramkan batin konsumen dalam mengkonsumsinya. Kesinambungan proses produksi halal dijamin oleh produsen dengan cara menerapkan Sistem Jaminan Halal dan saat ini kewenangan MUI dalam menerbitkan sertifikasi halal telah di pindah ke sebuah badan khusus dibawah naugan kementrian agama yaitu Badan Penyelengara Jaminan Produk Halal (BPJPH).

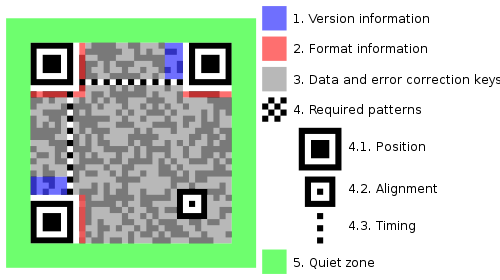
* 1. **Algoritma**

Menurut Rinaldi Munir (2005 : 176), Algoritma adalah urutan logis langkah-langkah penyeleseian masalah yang disusun secara sistematis. Alur pemikiran dalam menyelesaikan suatu pekerjaan yang dituangkan secara tertulis. Yang ditekankan pertama adalah alur pikiran, sehingga algoritma seseorang dapat juga berbeda dari algoritma orang lain. Sedangkan penekanan kedua adalah tertulis, yang artinya dapat berupa kalimat, gambar, atau tabel tertentu. Algoritma dapat dituliskan dalam berbagai notasi, misalnya dalam notasi kalimat-kalimat deskriptif. Dengan notasi kalimat deskriptif, deskripsi setiap langkah dijelaskan dengan bahasa sehari-hari secara jelas. Setiap langkah biasanya diawali dengan kata kerja seperti ‘baca’, ‘hitung’, ‘masukan’, ‘bagi’, ‘ganti’, dan sebagainya. Sedangkan pernyataan bersyarat dinyaakan dengan ‘jika’,’maka’, dan sebagainya

* 1. **Quick Response Code (QR - Code)**

Qr Code Adalah suatu jenis kode matriks atau kode batang dua dimensi yang dikembangkan oleh *Denso Wave,* sebuah divisi *Denso Corporation* yang merupakan perusahaan jepang dan di publikasikan pada tahun 1994 dengan fungsionalitas utama yaitu dapat dengan mudah dibaca oleh pemindai. QR Code merupakan kependekan dari *Quick Response Code* atau respon cepat, yang sesuai dengan tujuanya adalah untuk mmenyampaikan informasi dengan cepat dan mendapatkan respon yang cepat pula.berbeda dengan kode batang, yang hanya menyimpan informasi secara horizontal dan vertikal, oleh karena itu secara otomatis kode Qr dapat menampung informasi lebih banyak daripada kode batang biasa (Ida Bagus Putu Wirajaya Kusuma, 2012).

Penggunaan kode QR sudah sangat lazim di jepang, hal ini dikarenakan kemampuanya menyimpan data yang lebih besar dari kode batang, sehingga mampu mengkodekan informasi dalam bahasa jepang, sehingga dapat menampung huruf kanji. Kode QR telah mendapatkan standarisasi internasional dan standarisasi dari jepang berupa ISO/IEC18004 dan JIS-X-0510 dan telah digunakan secara luas melalui ponsel di jepang. (Ida Bagus Putu Wijaya Kusuma, 2012 ).

 Kode QR memiliki 5 bagian yang masing – masing mempunyai arti dan tugas masing – masing,berikut gambar di bawah marupakan bagian – bagian dan fungsi dari sebuah kode QR :

Gambar 2. 1 Bagian Bagian QR Code

Kode QR dapat menangani berbagai macam data termasuk angka, karakter abjad, huruf kanji, kana, hiragana,symbol, biner dan kode kontrol dengan maksimal sebanyak 7.089 karakter yang dapat di kodekan dalam satu kode QR.

Berikut tabel rincian kapasitas data kode QR

Tabel 2.1 kapasitas data QR Code

|  |  |
| --- | --- |
| **Kapasitas data Qr Code** | |
| Numerik | Max. 7.089 karakter |
| Alfanumerik | Max 4.296 karakter |
| Biner (8 bit) | Max. 2.953 byte |
| Kanji, full with Kana | Max. 1.817 Karater |

QR kode telah menjadi pupuler daripada kode Batang, hal ini membuat penggunaan dan keragaman QR code jauh lebih menarik daripada bar code, berikut beberapa kelebihan QR code :

1. QR Code menyimpan data hingga 7.089 karakter
2. QR Code memiliki kapasitas tinggi dalam pengkodean, yaitu mampu menyimpan semua jenis data seperti data numeri, data alphabetis, alfanumerik, kana, kanji, hiragana, simbol-simbol dan kode biner.
3. QR code merupakan kumpulan batang dengan rentang secara horizontal dan vertikal. Sehingga tidak membutuhkan space yang terlalu besar dalam pencetakan.
4. Secara statistik QR Code mampu melakukan *encoding* jumlah data yang sama disekitar ruang sepersepuluh dari barcode tradisional. Seperti menit QR Code disebut Micro *QR-Codes.*
5. Secara fitur dari QR Code kita tidak perlu memindai dari sudut tertentu. QR Code mampu di baca omnidirectional(360o) dengan kecepatan tinggi membaca. QR – Code scanner mampu menentukan cara yang benar untuk memecahkan kode isi dalam kode QR karena tiga kotak khusus yang ditempatkan di sudut – sudut simbol.
6. QR Code juga tahan terhadap kerusakan, sebab QR Code mampu memperbaiki kesalahan sampai 30 %. Oleh karena itu walaupun sebagian simbol QR Code kotor ataupun rusak,data tetap dapat disimpan dan dibaca.
   1. **Kriptografi**

Secara umum kriptografi adalah ilmu yang mempelajari bagaimana melakukan enkripsi dan deskripsi, dengan memanfaatkan model matematika tertentu. Dengan memanfaatkan kriptografi, maka kita dapat menyimpan data rahasia pada suatu penyimpanan tertentu. (Dian Rakyat, 2008).

Ilmu kriptografi sebenarnya sudah lama dipakai oleh orang jaman dulu, dalam sejarah beberapa kerajaan yang akan mengirim pesan ke kerajaan lainya isi dari pesan tersebut diubah sedemikian rupa sehingga orang lain akan kesulitan untuk membacanya, hanya penerima saja yang dapat membaca isi pesan tersebut.

Dari yang dicontohkan, penerima pesan membaca isi dari informasi tersebut dikarenakan penerima mengetahui bagaimana membacanya, atau bisa dibilang si penerima mempunyai “kunci (key)” untuk membuka isi dari pesan tersebut. Kunci disini dalam bentuk yang sama, baik untuk pengiriman maupun penerima.

Tujuan utama dari kriptografi yang ingin dicapai antara lain :

1. *Security,* informasi akan aman. Pesan yang dikirimkan tetap rahasia dan tidak diketahui oeh pihak lain, umumnya hal ini dilakukan dengan cara membuat suatu algoritma matematik yang mampu mengubah data sehingga data sulit untuk dibaca dan dipahami.
2. *Integrity,* informasi tidak rusak. Mampu mengenali /mengdeteksi adanya manipulasi (Penghapusan, Pengubahan atau penambahan) data yang tidak sah oleh pihak lain.
3. *Authentication,* informasi hanya akan diakses oleh pihak yang berhak. Mampu melakukan identifikasi, baik otentifikasi pihak-pihak yang terlibat dalam pengiriman data maupun otentifikasi keaslian data/informasi.
4. *Non-repudiation,* informasi yang sekali dikirim. Artinya, pesan hanya sekali dikirim sehingga dapat menyangkal pihak lain yang bukan pengirim pesan.

Dalam kriptografi ada beberapa istilah- istilah yang sering digunakan :

1. *Plaintext,* semua data yang belum diproses melalui suatu algoritma kroptografi yang biasa berupa text, gambar ataupun bentuk lain.
2. *Chipertext,* adalah keluaran algoritma enkripsi.
3. Enkripsi (*Encryption*), proses pengubahan dari *plaintext* ke *chipertext* dengan algoritma kriptografi tertentu.
4. Deskripsi (*Descryption*), proses pengubahan dari *Chipertext* ke  *plaintext* dengan algoritma kriptografi tertentu.
5. *Chiper,* suatu algoritma kriptografi tertentu yang dapat melakukan proses enkripsi dan deskripsi.
6. *Key,* parameter kunci yang digunakan oleh chipper dalam melakukan proses enkripsi dan deskripsi.
7. *Message Digest,* ukuran *byte* kecil yang menunjukan dari ukuran *plaintext.*
8. *Hash,* nama lan dari *Message Digest.*
   1. **Kriptografi Modern**

Kriptografi modern berorietasi bit sebab penyandian modern mengguanakan media computer untuk mengolah pesan. Pesan pada sandi modern tidak selalu berupa rangkaian karakter bisa saja berupa rangkaian bit seperti berkas video atau gambar..

Algoritma kriptografi modern terdiri dari dua bagian yaitu :

1. Kunci Simetri

Proses enkripsi dan deskripsi data secara keseluruhan menggunakan kunci yang sama. Skema ini berdasarkan jumlah data perproses dan alur pengolahan data didalamnya dibedakan menjadi dua kelas yaitu blok-chiper dan stream-chiper. Berikut merupakan contoh yang menggunakan kunci simetri antara lain :

1. *Data Encryption Standard* (DES)
2. *Advance Encryption Standard* (AES)
3. *International Data Encryption Algorithm* (IDEA)
4. RCA4
5. Kunci Asimetris

Kunci asimetris menggunakan kunci yang berbeda dalam setiap proses enkripsi dan deskripsinya. Kunci ini sering di sebut dengan kunci public (*Public - Key*) karena kunci enkripsinya dibuat supaya diketahui oleh orang lain dalam melakukan enkripsinya, dalam tahap deskripsinya kunci yang digunakan hanya diketahui oleh yang berhak memiliki kunci rahasia.

Dari beberapa algoritma yang ada dalam kriptografi, yang penulis gunakan dalam penelitian ini menggunakan algoritma AES 128

* 1. **Algoritma *Advance Encryption Standard* (AES)**
     1. **Sejarah AES**

Pada yahun 90-han, setelah beberapa tahun standar penyandian simetris DES dianggap tidak lagi aman, lembaga standar Amerika Serikat *National Institute of Standard and Technology* (NIST) membuat sayembara untuk menggantikan DES dengan sebuah sistem penyandian yang disebut Advance Encryption Standard (AES) pada tanggal 12 September 1997. NIST memberikan spesifikasi AES, yaitu harus memiliki panjang blok 128 bit dan mampu mendukung panjang kunci 128, 192 dan 256.

Setelah beberapa seleksi, NIST memilih sistem penyandian *Rijdael* yang dikembangkan oleh Joan Daemen dan Vincen Rijment sebagai sistem penyandian AES pada taun 2000, pemilihan Rijdael berdasarkan kriteria :

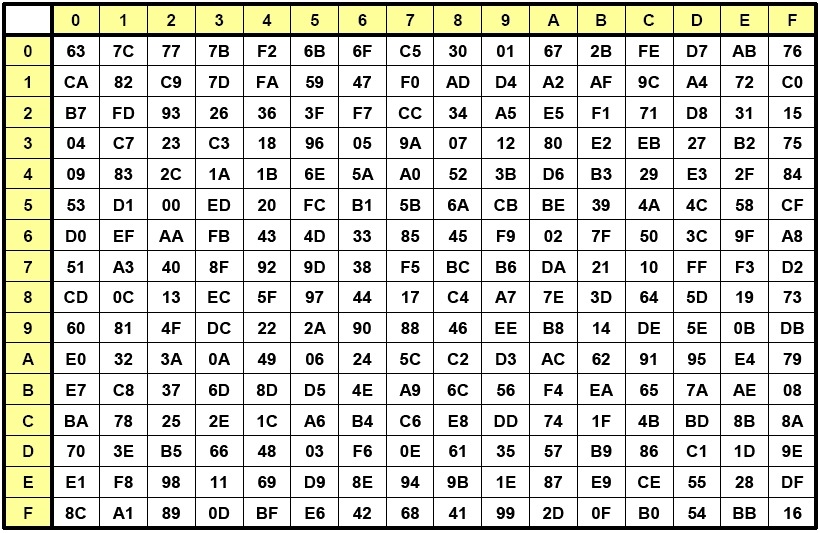
1. Keamanan. Sistem penyandian harus tahan terhadap serangan analisis sandi dan serangan secara *brute force.*
2. Biaya. Sistem penyandian harus memiliki biaya komputasi dan memori yang efisien sehingga dapat diimplementasikan secara perangkat keras maupun perangkat lunak.
3. Karakteristik algoritma dan implementasi. Sistem penyandian harus bersifat terbuka, fleksibel dan sederhana.

Pada tahun 2001 akhirnya NIST mempublikasikan AES sebagai standar pemrosesan dokumen pada dokumen FIPS-PUB 197, (NIST, 2001). (Rifki sadikin, 2012)

* + 1. **Enkripsi AES**
       1. **Transformasi *SubBytes***

AES menggunakan subtitusi nonlinear pada ukuran *byte* yang akan disebut *SubByte.* Setiap element pada state dari elemen *S0,0*sampai *S3,3* dikarenakan *SubByte* menggunakan tabel substitusi S-Box. Tabel S-Box yang digunakan adalah :

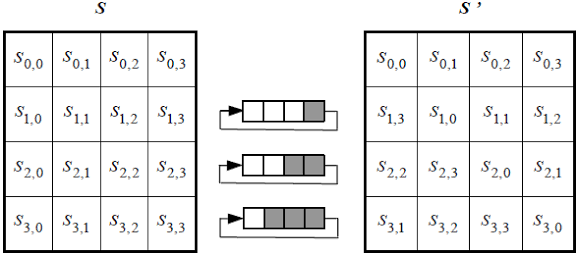
**Tabel 2.2** S-Box*Rijdael* xy bilangan hexadesimal



Transformasi *SubBytes* dapet menggunakan tabel subtitusi, yaitu dengan cara meninterpretasikan *byte* masukan *S­i,j*sebagai bilangan hexadecimal, kemudian digit kiri menunjukan indeks baris dan digit kanan menunjukan indeks kolom di tabel substitusi. Nilai *byte*  pada tabel substitusi yang dirujuk oleh indeks baris dan kolom menjadi nilai yang mensubstitusi *Si,j.* tabel substitusi untuk SubBytes diberikan oleh tabel 2.2 diatas.(Rifki Sadikin, 2012).

* + - 1. **Transformasi *ShiftRows***

Transformasi *ShiftRows* merupakan jenis transformasi permutasi, yaitu pengubahan posisi elemen pada *state* tanpa mengubah nilainya. Trasnformasi *shiftrows* melakukan pergeseran secara *wrapping*  (siklik) pada 3 baris terakhir *array state.* Jumlah pergeseran bergantung pada niai baris (*r*). Baris *r* = 1 digeser sejauh 1 *byte,* baris *r* = 2 digeser sejauh 2 *byte,* dan baris *r = 3* digeser sejauh 3 byte, *r=0* tidak digeser.



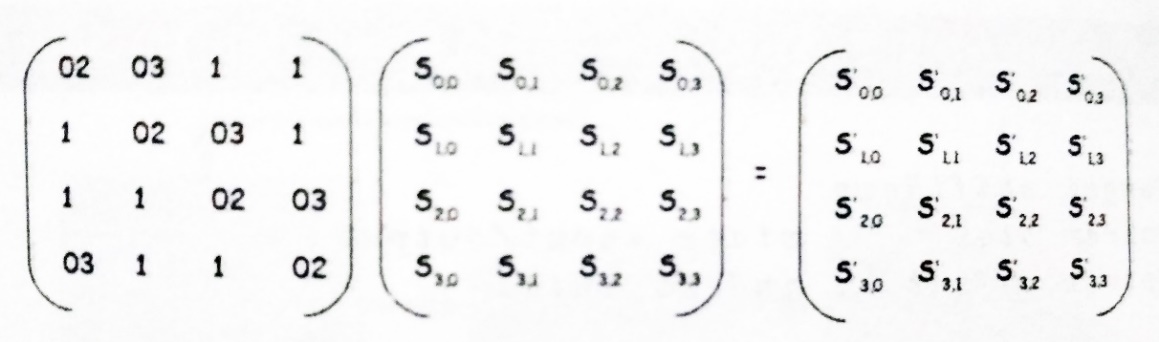
**Gambar 2.2** Transformasi *ShiftRows*

* + - 1. **Transformasi *MixColums***

Transformasi MixColums mengalikan setiap colom *state* dengan polinom a(x) mod (x4 + 1), *MixColumns* memberikan difusi pada *chipper*. Setiap kolom diperlakukan sebagai polinom 4 suku pada GF(28). Polinom a (x) yang di tetapkan adalah :

a(*x*) = {03}*x*3 + {01}*x2* + {02}

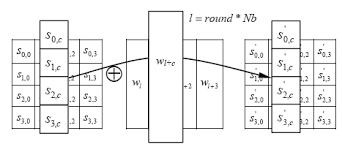
Transformasi ini dinyatakan sebagai perkalian matriks pada gambar dibawah.

Gambar 2.3 Transformasi MixColumns

* + - 1. **Transformasi AddRoundKey**

Transformasi *AddRoundKey* mencampur sebuah *state*memasukan dengan dengan kunci ronde dengan operasi eksklusif OR. Setiap elemen kunci state masukan yang merupakan sebuah byte dikarenakan operasi eksklusif OR dengan byte pada posisi yang sama di kunci ronde (kunci ronde di representasikan sebagai *state*).

Transformasi AddRoundKey merupakan transformasi yang bersifat *self invers***,** yaitu trasnformasi invers sama dengan transformasi aslinya asalkan menggunakan kunci ronde yang sama, gambar di bawah mengilustrasikan transformasi AddRoundKey.



Gambar 2.4Transformasi AddRoundKey

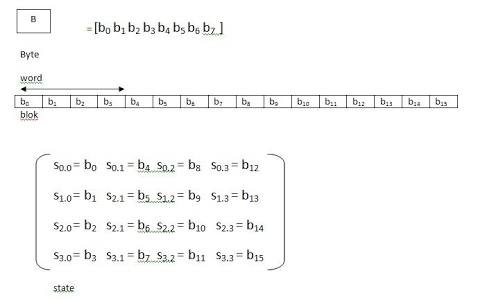
* + 1. **Deskripsi AES**

AES merupakan sistem penyandian blok yang bersifat non-Feistel karena AES menggunakan komponen yang selalu memiliki invers dengan panjang blok 128, 192 dan 256 bit. Penyandian AES menggunakan proses yang berulang – ulang yang disebut dengan ronde (*round*). Jumlah ronde yang digunakan oleh AES tergantung dengan panjang kunci yang digunakan. Setiap ronde membutuhkan kunci ronde dan masukan dari ronde berikutnya. Kunci ronde dibangkitkan berdasarkan berdasarkan kunci yang diberikan. Relasi antara jumlah ronde dan panjang kunci diberikan oleh tabel 2.1(Rifki Sadikin, 2012)

Tabel 2.3 Hubungan antara jumlah ronde dan panjang kunci AES

|  |  |
| --- | --- |
| Panjang Kunci AES (bit) | Jumah Ronde (Nr) |
| 128  192  256 | 10  12  14 |

AES menggunakan 5 unit data : *bit, byte, word,* blok dan *state.* Bit merupakan satuan data terkecil, yaitu nilai digit sistem biner. Sedangkan *byte*  berukuran 8 bit, *word* berukuran 4 *byte* (32 bit), blok berukuran 16 byte (128 bit) sedangkan *state* adalah blok yang ditata sebagai matriks 4x4



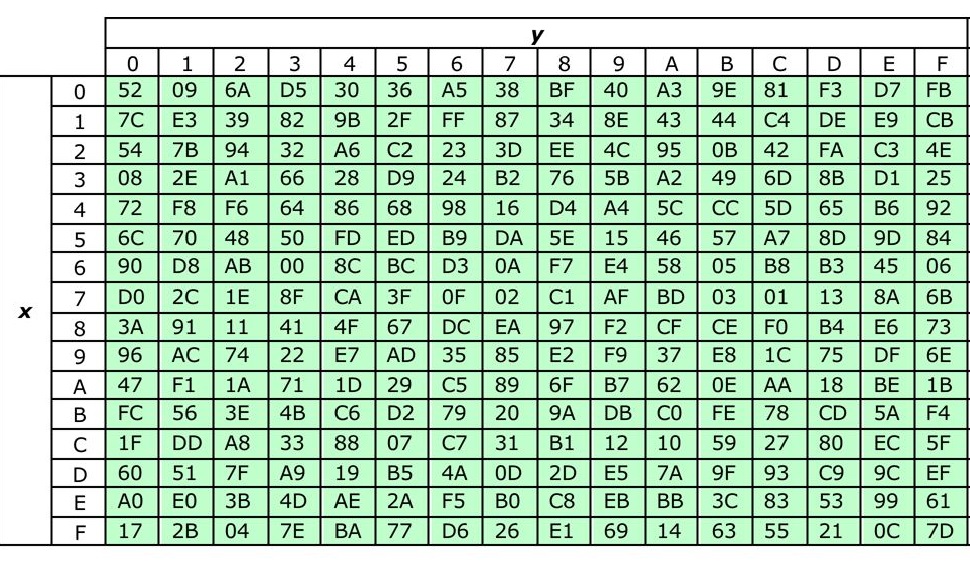
**Gambar 2.5** Unit Data AES

Pada dasarnya algoritma deskripsi pada AES merupakan kebalikan dari algoritma enkripsi AES (*invers*). Algoritma deskripsi AES menggunakan transformasi invers pada semua trasnformasi dasar AES memiliki transformasi invers, yaitu ***InvSubBytes, InvShiftRows*** dan ***InveMixColumnns.*** Pada proses trasnformasi ***AddRoundKey*** proses deskripsi yang dilakukan sama dengan proses enskripsi karena trasnformasi yang bersifat self-invers dengan syarat menggunakan kunci yang sama.

* + - 1. **Transformasi InvSubBytes**

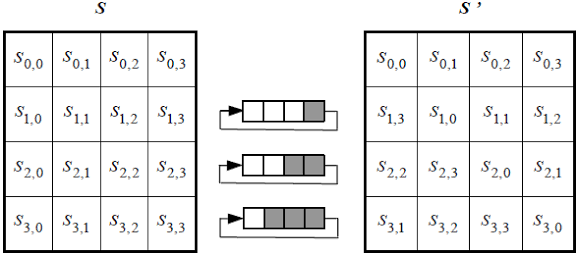
InvSubByte merupakan invers dari transformasi SubBytes yang mana invers pada proses InvSubBytes terdapat pada setiap byte dalam state dalam S-Box, berikut tabel substitusi InvSubBytes :

Tabel 2.4 Tabel InvSubBytes S-Box xy bilangan Hexadesimal



* + - 1. **Transformasi InvShiftRow**

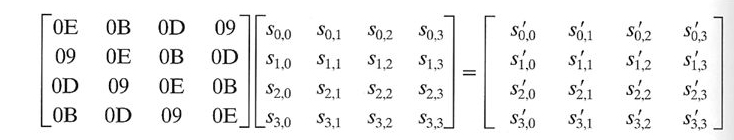
Transformasi invers terhadap Shiftrows disebut InvShiftRows. Transformasi InvShitRows terhadap sebuah state menggunakan oprasi (*circular shift right*) pada setiap barisnya yang banyak gesernya sesuai dengan indeks baris, berikut gambar proses transformasi *InvShiftRows.*



Gambar 2.6 Transformasi *InvShifRows*

* + - 1. **Transformasi InvMixColumnns**

InversTransformasi MixColumns adalah transformasi InvMixColumns. Transformasi InvMixColumns menggunakan perkalian matriks antara sebuah konstans dengan state. Konstan yang dipakai invMixColumns merupakan invers matriks konstan pada transformasi MixColums, ganbar 2.6 dibawah merupakan perkalian matriks yang dipakai oleh transformasi InvMixColums. Proses komputasi pada InvMixColums meggunakan operator perkalian dan penjumlahan GF(28).



Gambar 2.7 Transformasi InvMixColums

* 1. **Flowchart**

*Flowchart* (Bagan Alir) adalah suatu bagan yang menggambarkan atau mempresentasikan suatu algoritma atau prosedur untuk menyelesaikan masalah. Flowchart terbagi menjadi dua, flowchart sistem yaitu bagan yang menggambarkan suatu prosedur dan proses suatu file dalam suatu media menjadi file dalam media yang lain dalam suatu sistem data. Sedangkan flowchart program yaitu bagan yang menggambarkan urutan logika dari suatu prosedur pemecahan masalah. (Heri Sismoro, 2005).

Tabel 2.5 Simbol-Simbol Dalam Flowchart

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
|  | **Terminator**, simbol untuk menunjukkan awal atau akhir dari aliran proses**.** |
|  | **Processing, menujukan pengolahan aritmatika dan pemindahan data.** |
|  | **Preparation, memberikan nilai awal pada suatu variabel atau counter.** |
|  | *Connector*, tanda panah yang menunjukkan arah aliran dari satu proses ke proses yang lain. |
|  | **Decision,** menunjukan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban ya / tidak. |
|  | **Input/Output***,* simbol untuk menunjukkan data yang menjadi input  atau output proses. |
|  | **Sub-process***,* simbol untuk menunjukkan bahwa dalam langkah yang dimaksud terdapat flowchart lain yang menggambarkan langkah tersebut lebih  rinci. |
|  | **Connector, penghubung pada halaman yang sama.** |
|  | **OOff page connector, penghubung pada halaman yang berbeda** |

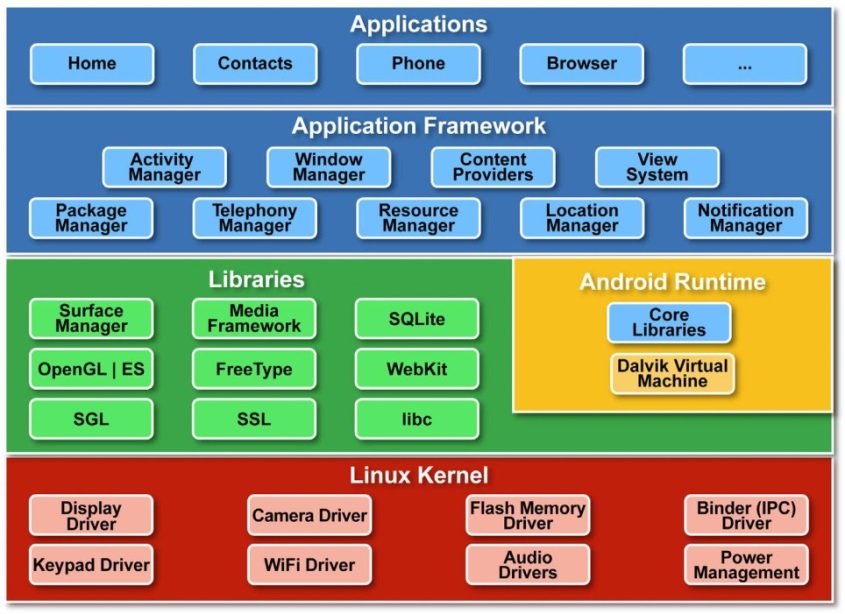
* 1. **Bahasa Pemrograman Java**

Java adalah bahasa pemrograman serbaguna. Java dapat digunakan untuk membuat suatu program sebagaimana membuatnya seperti bahasa lain. Java juga mendukung sumberdaya internet yang sangat populer yaitu WEB, selian itu java juga mendukung aplikasi klien-server dalam jaringan lokal dan luas. (Abdul Kadir:2004).

Java dikembangkan oleh sun microsystem pada agustus 1991 dengan nama semula *OAK***.** *OAK* adalah sejenis pohon jati yang terlihat dari jendela tempat pembuatnya, james Gosling bekerja. Ada yang mengatakan oak adalah singkatan dari “*Object Application Kernel”.* pada januari 1995 karena nama Oak dianggap kuranh komersial maka diganti dengan nama Java.

Dalam sejumlah literatur disebutkan bahwa java merupakan hasil perpaduan sifat dari sejumlah bahasa pemrograman yaitu C, C++, Object-C, SmallTalk, dan CommonLISP. Selain itu java juga dilengkapi dengan unsur keamanan. Yang tak kalah penting java menambahkan paradigma pemrograman yang sederhana. Jika anda telah mengenal C atau C++ yang mengandalkan pointer dan merasakan keruwetannya, java justru meninggalkan sehingga memperoleh kemudahaan java saat menggunakannya. Java memiliki kelebihan, sebagai berikut :

1. *Multiplatform*. Kelebihan utama dari Java ialah dapat dijalankan di beberapa *platform* / sistem operasi komputer, sesuai dengan prinsip *tulis sekali, jalankan di mana saja*. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / *bytecode*) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas operating system Linux tetapi dijalankan dengan baik di atas Microsoft Windows.
2. OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek) yang artinya semua aspek yang terdapat di Java adalah Objek. Java merupakan salah satu bahasa pemrograman berbasis oebjek secara murni. Semua tipe data diturunkan dari kelas dasar yang disebut *Object*. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir.
3. Perpustakaan Kelas Yang Lengkap, Java terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya.
4. BergayaC++, memiliki sintaks seperti bahasa pemrograman [C++] sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java.
   1. **Android**
      1. **Sejarah Singkat Android**

Android adalah sebuah sistem operasi utuk perangkat mobile berbasis linux yang mencangkup sistem operasi, *middleware* dan aplikasi. Awalnya, Google Inc. membeli android Inc. yang merupakan pendatang baru yang membuat piranti lunak untuk *smartphone*. kemudian untuk mengembangkan android, dibentuklah *Open* *Handset Alliance* (OHA).

Gambar 2.8 Arsitektur Android

Sejak tahun 2008, Android secara bertahap telah melakukan sejumlah pembaruan untuk meningkatkan kinerja sistem operasi, menambahkan fitur baru, dan memperbaiki bug yang terdapat pada versi sebelumnya. Setiap versi utama yang dirilis dinamakan secara alfabetis berdasarkan nama-nama makanan pencuci mulut atau cemilan bergula; misalnya, versi 1.5 bernama Cupcake, yang kemudian diikuti oleh versi 1.6 Donut. Versi terbaru adalah 5.0 Lollipop, yang dirilis pada 15 Oktober 2014. (**Nazruddin Safaat H.2012**).

* + 1. **Arsitektur Android**

Secara garis besar arsitektur android dapat dijelaskan dan digambarkan sebagai berikut (**Nazruddin Safaat H.2012**):

Gambar 2.8 arsitektur sistem operasi android (Nazruddin Safaat H.2012)

1. *Application and Widgets*

*Aplication* dan *widget* ini adalah *layer* dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita download aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Dilayer ini terdapat aplikasi inti termasuk email, program SMS, kalender, peta, browser, kontak, dan lain-lain

1. *Application Frameworks*

Android adalah “*Open Development Platform*” yaitu android menawarkan kepada pengembang untuk memeberi kemampuan kepada pengembang unuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *services* *background*, mengatur alarm dan menambahkan notifikasi, dan sebagainya.

1. *Libraries*

*Libraries* ini adalah *layer* dimana fitur fitur android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan diatas *kernel*, *layer* ini meliputi berbagai *library* c/c++ inti seperti libc dan SSL, serta :

1. *Libraries* media untuk pemutaran media audio dan video.
2. *Libraries* untuk manajemen tampilan.
3. *Libraries* graphics mencangkup SGL dan OpenGL untuk grafis 2D dan 3D.
4. *Libraries* SQLite untuk dukungan database.
5. *Libraries* SSL dan *Webkit* terintegrasi dengan *web* *browser* dan *security*.
6. *Libraries* *LiveWebcore* mencangkup modern web browser dengan *engine embedded web view.*
7. *Android Run Time*

*Layer* yang membuat aplikasi android dapat berjalan dimana saja dalam prosesnya menggunakan implementasi linux. Didalam *Android* *Run* *Time* terbagi menjadi dua bagian yaitu :

1. *Core* *Libraries*: aplikasi yang dibangun dalam bahasa java, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa *Java/C yang ditangani oleh core libraries.*
2. *Dalvik Virtual Machine* : virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi fungsi secara efisien, untuk melakukan *threading* dan manajemen tingkat rendah.
3. *Linux Kernel*

*Linux kernel* adalah *layer* dimana inti dari *operating system* dari android itu berada. Berisi file-file sistem yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem-sistem operasi android lainnya.

* 1. **Android Studio**

Android Studio sebuah software khusu yang didesain di Lingkungan Pengembangan Terpadu - *Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android, yang di rilis oleh  [IntelliJ IDEA](https://www.jetbrains.com/idea/). Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android, antara lain (developer.android.com) :

1. Sistem versi berbasis *Gradle* yang fleksibel
2. Emulator yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android
4. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
6. Alat pengujian dan kerangka kerja yang ekstensif
7. Alat *Lint* untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain
8. Dukungan C++ dan NDK
9. Dukungan bawaan untuk [*Google Cloud Platform*](http://developers.google.com/cloud/devtools/android_studio_templates/?hl=id), mempermudah pengintegrasian *Google Cloud Messaging dan App Engine*

Setiap proyek di Android Studio berisi satu atau lebih modul dengan file kode sumber dan file sumber daya. Jenis modul meliputi:

1. Modul aplikasi Android
2. Modul perpustakaan
3. Modul Google App EngineSecara default,

Android Studio menampilkan file proyek Anda dalam tampilan proyek Android, seperti yang ditunjukkan pada gambar di bawah Tampilan ini disusun oleh modul untuk menyediakan akses cepat ke file sumber utama proyek Anda.Semua file build terlihat di tingkat atas di bawah *Gradle Scripts* dan setiap modul aplikasi berisi folder berikut:

Manifest: Berisi file AndroidManifest.xml.

1. java: Berisi file kode sumber Java, termasuk kode uji JUnit.
2. res: Berisi semua sumber daya non-kode, seperti tata letak XML, string UI, dan gambar bitmap.

Berikut tampilan antar muka pengguna Android studio (developer.android.com) :

Gambar 2.9 Tampilan antar muka Android Studio

1. ***Toolbar*** memungkinkan Anda untuk melakukan berbagai jenis tindakan, termasuk menjalankan aplikasi dan meluncurkan alat Android.
2. ***Navigation Bar*** membantu Anda bernavigasi di antara proyek dan membuka file untuk diedit. Bagian ini memberikan tampilan struktur yang terlihat lebih ringkas dalam jendela Project.
3. ***Windows Editor*** adalah tempat Anda membuat dan memodifikasi kode. Bergantung pada jenis file saat ini, editor dapat berubah. Misalnya, ketika melihat file tata letak, editor menampilkan Layout Editor.
4. ***Tool Window Bar*** muncul di luar jendela IDE dan berisi tombol yang memungkinkan Anda meluaskan atau menciutkan jendela alat individual.
5. ***Tool Window*** memberi Anda akses ke tugas tertentu seperti pengelolaan proyek, penelusuran, kontrol versi, dan banyak lagi, bagian bisa meluaskan dan juga menciutkannya.
6. ***Status Bar*** menampilkan status proyek Anda dan IDE itu sendiri, serta setiap peringatan atau pesan.
   1. **Xampp**

Menurut Bunafit Nugroho (2008 : 2) XAMPP adalah suatu bundle *web server* yang populer dan biasa digunakan di *windows* karena kemudahan instalasinya. Bundle program *open source* tersebut berisi antara lain server web Apache, interpreter PHP, dan basis data MySQL. Setelah menginstal XAMPP kita bisa memulai pemerograman PHP di computer sendiri maupun mencoba menginstall aplikasi yang berbasis web

* 1. **PHP**

PHP adalah kependekan dari “ *Hypertext Preprocessor*”, yaitu Bahasa pemrograman universal untuk penanganan, pembuatan dan pengembangan sebuah situs web dan bisa digunakan bersamaan dengan HTML. PHP banyak dipakai untuk membuat situs web dinamis.

PHP merupakan sekumpulan skrip atau Bahasa program yang memiliki fungsi utama yaitu mampu mengumpulkan dan mengevaluasi hasil survey atau bentuk apapun ke *server* *database* dan pada tahap selanjutnya akan menciptakan efek beruntun. Efek beruntun ini berupa tindakan dari skrip lain yang akan melakukan komunikasi dengan *database*, mengumpulkan dan mengelompokan informasi.(Mundzir M.F, 2014).

* 1. **MySQL**

Mysql merupakan *softaware database* berbasis *command form (shell / console),* dengan menggunakan sql (*Structure Query Language*) yaitu Bahasa standar yang digunakan dalam *database,* karena hampir semua software database mendukung SQL. (Aib.V.Dian Sono, 2005).

SQL adalah bahasa standar yang digunakan untuk mengakses database server. Bahasa ini pada awalnya dikebangkan oleh IBM, namun telah diadopsi dan digunakan sebagai standar industry. Dengan menggunakan SQL,proses akses database menjadi lebih *user-friendly* dibanding dengan menggunakan dBase dan Clipper yang masih menggunakan perintah – perintah pemrograman (Madcoms, 2009).

* 1. **HTML**

HTML (Hypertext Markup Language) merupakan salah satu format yang digunakan dalam pembuatan dokumen dan aplikasi yang berjalan di halaman web. HTML merupakan protocol yang digunakan untuk mentransfer data antara web server ke web browser. Protokol ini mentransfer dokumen –dokumen web yang ditulis atau berformat HTML.

Dikatakan *markup* languagekarena HTML berfungsi untuk menformat file document teks biasa untuk bisa ditampilkan pada web browser dengan bantuan tanda – tanda yang sudah ditentukan. Hal tersebut dapat dilakukan dengan menambahkan element atau yang sering kita sebut dengan *tag(<* dan *>).* (Bimo Sunarfrihantono,2003).

* 1. ***Web Browser***

*Web Browser*adalahsoftware yang digunakan untuk menampilkan informasi dari server web. Software ini telah dikembangkan dengan menggunakan user *interface* grafis, sehingga pemakai dengan melakukan point dan clik dapat pindah antar dokumen. (Betha S, Husni I, 2010:5).

* 1. **Rational Unified Process**

Menurut Rosa dan Shalahudin (2016:125), *Rational Unified Process* (RUP) adalah metodologi pengembangan perangkat lunak yang diformulasikan oleh *Rational Software Corporation* (sekarang menjadi salah satu divisi IBM), yang menggunakan UML (*Unified Modeling Language*) sebagai bahasa pemodelan selama periode pengembangan yang dilakukan berulang – ulang (*iterative incremental)* sebagai model siklus pengembangan perangkat lunak. Model ini membagi suatu sistem aplikasi menjadi beberapa komponen sistem dan memungkinkan para pengembang aplikasi untuk menerapkan metoda *iterative* (analisis, disain, implementasi dan pengujian) pada tiap komponen.

Berikut proses pengulangan / iterative pada RUP secara global dapat dilihat pada gambar berikut :

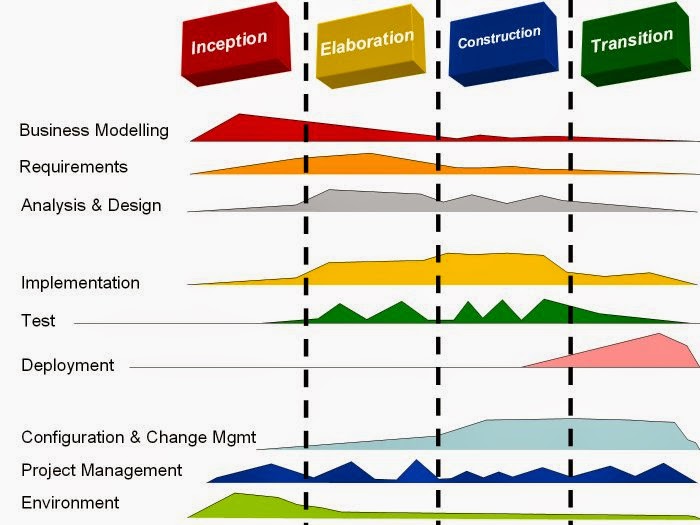


Gambar 2.10Proses Iteratif RUP

RUP menggunakan konsep *object oriented*, dengan aktifitas yang berfokus pada pengembangan model dengan menggunakan *Unified Model Language* (UML). Melalui gambar dibawah dapat dilihat bahwa RUP memiliki, yaitu:

1. Dimensi pertama digambarkan secara horizontal. Dimensi ini mewakili aspek-aspek dinamis dari pengembangan perangkat lunak. Aspek ini dijabarkan dalam tahapan pengembangan atau fase. Setiap fase akan memiliki suatu *major milestone* yang menandakan akhir dari awal dari phase selanjutnya. Setiap phase dapat berdiri dari satu beberapa iterasi. Dimensi ini terdiri atas *Inception*, *Elaboration*, *Construction*, dan *Transition*.
2. Dimensi kedua digambarkan secara vertikal. Dimensi ini mewakili aspek-aspek statis dari proses pengembangan perangkat lunak yang dikelompokkan ke dalam beberapa disiplin. Proses pengembangan perangkat lunak yang dijelaskan kedalam beberapa disiplin terdiri dari empat elemen penting, yakni *who is doing*, *what*, *how* dan *when*. Dimensi ini terdiri atas *Business Modeling, Requirement, Analysis and* Design*, Implementation, Test, Deployment, Configuration* dan *Change Manegement, Project Management, Environtment.* Berikut Penjelasanya :
3. *Business Modeling* Pada tahap ini, terdapat identifikasi dan deskripsi langsung dari area dan permasalahan untuk *redesign* atau *reengineering*, *beserta struktur dan proses–proses bisnis organisasi.*
4. *Requirements* Tujuan utama pada fase ini adalah menyusun sistem apa yang seharusnya ada dan mengapa perlu dibuat, mendefinisikan batas dari sistem, melihat kemungkinan ancaman keamanan serta bagaimana cara penanggulangannya, dan mengestimasi biaya dan skala waktu yang rumit. Isi dari sistem dibangun yang kemudian diterjemahkan kedalam *use case* model dengan tambahan spesifikasi kebutuhan. Baik kebutuhan fungsional dan nonfungsional akan dikumpulkan dan dianalisis. Kebutuhan user dan stakeholder serta fitur high-level didefinisikan dan kemudian diubah menjadi *specific software requirements*.
5. *Analysis and Design* Pada fase ini, semua requirement pada tahap kedua akan diubah menjadi spesifikasi implementasi.
6. *Implementation* Pada tahap ini, semua analisa dan desain yang telah dibuat pada fase sebelumnya akan diimplementasikan dan diterjemahkan menjadi *kode program.*
7. *Testing Pada tahap ini, pengembang software akan menguji dan memverifikasi semua interaksi komponen, kebutuhan yang telah diimplementasikan dan kualitas dari software yang telah dikembangkan.*
8. *Deployment Pada tahap ini, pengembang software menyebarkan software yang telah selesai kepada user. Pengembang software juga menyediakan dokumentasi untuk semua fitur dan fungsi. Pada tahap ini juga, pengembang software mendapatkan umpan balik dan masukan terhadap software yang berujung pada modifikasi fungsi dan fitur agar menjadi lebih baik.*
9. *Configuration and Change Management Tahap ini menjalankan dan* merawat integritas dari proyek. Kegiatannya meliputi monitoring dan mengatur perubahan permintaan, perubahan biaya, dan tetap mengontrol berbagai versi produk. Tahap ini juga meliputi manajemen konfigurasi hardware dan software.
10. *Project Management Tahap ini menyediakan framework untuk mengatur software dan resiko. Tahap ini juga menyediakan pedoman untuk planning, staffing, monitoring dan secara umum menunjukan manajemen proyek.*
11. *Enviroment Tahap ini menjelaskan tentang infrastruktur dan metode yang dibutuhkan untuk mengembangkan sistem*

Berikut gambar Arsitektur dari Metode Rational Unified Process



Gambar 2.11Arsitektur *Rational Unified Process.*

Pada gambar diatas terdapat 4 fase RUP berikut penjelasanya :

1. *Inception*

Pada tahap ini lebih memodelkan proses bisnis yang di butuhkan (*business modelling*) dan mendifinisikan kebutuhan sistem yang akan dibuat (*requirements*). Dan pada tahap ini penulis melakukan perancangan use case bussines yang menggambarkan interkasi antara sistem atau aplikasi e-halal dengan user, selain itu penulis juga mengggambarkan *requirement workflow* yang menggambarkan sistem apa yang dilakukan dalam bentuk *use case diagram*  aplikasi E-halal.

1. *Elaboration* (perluasan dan perencanaan)

Tahap untuk melakukan desain secara lengkap berdasarkan hasil analisis ditahap *inception.* Aktivitas yang dilakukan pada tahap ini antara lain mencakup pembuatan desain arsitektur subsistem (*architecture pattern*), desain komponen sistem, desain format data (protokol komunikasi), desain antarmuka/tampilan, desain peta aliran tampilan, penentuan *design pattern* yang digunakan, pemodelan diagram UML (diagram *activity, class*) dan pembuatan dokumentasi.

Pada tahap ini penulis melakukan perancangan *user interface*, *database*, denagn mengguanakan sebuah dokumen yaitu spesisfikasi kebutuhan perangkat lunak (SKPL). Kedepanya apliksi e-halal ini akan di gunakan dalam masysarakat umum untuk melakukan pengecekan kehalalan suatu produk berdasakan nomor sertifikasi yang terdaftar di BPJPH.

1. *Construction* (Konstruksi)

Tahap ini focus pada pengembangan komponen dan fitur – fitur sistem, implentasi kode – kode program dan pengujian peragkat lunak. Pada tahap ini penulis mulai melakukan penulisan kode – kode program (*coding)* degan menggunakan Bahasa pemrograman java untuk android (c*lient*), PHP untuk sisi web(server) dan MySQL untuk Pembuatan basis data.

1. *Transition*

Pada tahap ini penulis mulai menyerahkan sistem/aplikasi ke konsumen (*roll-out*), dilakuakan pelatihan, pemeliharaan, dan pengujian hasil akhir sisem, apakah sudah memenuhi kebutuhan user atau belum.

Akhir darike empat fase ini adalah produk perangkat lunak yang sudah lengkap. Keempat fase RUP ini dilakukan secara berurutan dan iterative dimana setiap interasi dapat digunakan untuk memperbaiki iterasi berikutya.

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

*Unified Modelling Language* atau biasa disingkat UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks- teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek **(AS Rosa dan M. Shalahuddin, 2013.137).**

* + 1. **Diagram UML**

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokan pada 3 kategori. Pembagian kategori dan macam macam diagram tersebut dapat dilihat pada gambar 2.15 berikut ini **(AS Rosa dan M. Shalahuddin, 2013.140)** :

Gambar 2.12 Diagram UML

Berikut adalah penjelasan singkat dari pembagian kategori tersebut :

* 1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan
  2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem
  3. *Intraction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain ataupun subsistem pada suatu sistem.

Berikut penjelasan dari diagram diagram yang terdapat pada *UML* :

1. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan model atau operasi. Dimana atribut adalah variabel variabel yang dimilik oleh suatu kelas dan metode atau operasi merupakan fungsi fungsi yang dimiliki oleh suatu kelas.

Kelas kelas yang ada pada struktur sistem harus dapat melakukan fungsi fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak dapat membeuat kelas kelas didalam program perangkat lunak sesuai dengan perancangan diagram kelas.

Berikut adalah simbol simbol yang ada pada diagram kelas :

Tabel 2.6 simbol pada diagram kelas

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| Kelas   |  | | --- | | Nama\_kelas | | +atribut | | +operasi() | | Kelas pada strutur sistem |
| Antarmuka / interface  Nama\_interface | Sama dengan konsep interface dalam pemrograman berorientasi objek |
| Asosiasi / association | Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity |
| Asisasi berarah / directed association | Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity |
| Generalisasi | Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus) |
| Kebergantungan / dependency | Kebergantungan antarkelas |
| Agregasi / aggregation | Relasi antarkelas dengan makna semua bagian (whole-part) |

1. *Object Diagram*

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggungjawabkan. Diagram objek juga berfungsi untuk mendefinisikan contoh nilai atau suatu isi dari struktur tiap kelas.

Hubungan link pada diagram objek merupakan hubungan memakai dan dipakai dimana dua buah objek akan dihubungkan oleh link jika ada objek yang dipakai oleh objek lainnya. Berikut adalah simbol simbol yang ada pad adiagram objek :

Tabel 2.7 simbol pada diagram objek

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| Objek   |  | | --- | | Nama\_objek : nama\_kelas | | Atribut = nilai | | Objek dari kelas yang berjalan saat sistem dijalankan |
| Link | Relasi antar objek |

1. *Component Diagram*

Diagram komponen dibuat untuk menunjukan organisasi dan ketergantungan dianara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada didala sistem. Komponen dasar yang biasanya ada pada suatu sistem adalah sebagai berikut :

* + 1. Komponen user interface yang menangani tampilan
    2. Komponen bussiness processing yang menangani fungsi fungsi proses bisnis
    3. Komponen data yang menangani manipulasi data
    4. Komponen security yang menangani keamanan system

Komponenlebih fokus pada penggolongan secara umum fungsi fungsi yang diperlukan dan berikut adalah simbol simbol yang ada pada diagram komponen :

Tabel 2.8 simbol pada diagram komponen (AS Rosa dan M. Shalahuddin, 2013.149)

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| Package | Package merupakan sebuah bungkusan dari satu atau lebih komponen |
| Komponen | Komponen sistem |
| dependency | Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai |
| Antarmuka / interface | Sama dengan konsep interface pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen |
| Link | Relasi antar komponen |

1. *Package Diagram*

Package diagram menyediakan cara mengumpulkan elemen elemen yang saling terkait dalam diagram UML. Hampir semua diagram dalam UML dapat dikelompokan dengan menggunakan package diagram. Berikut simbol simbol yang digunakan dalam package diagram :

Tabel 2.9 simbol pada diagram paket (AS Rosa dan M. Shalahuddin, 2013.153)

|  |  |
| --- | --- |
| **simbol** | **deskripsi** |
| Package | Package merupakan sebuah bungkusandari satu atau lebih kelas atau elemen diagram UML lainnya |
| Elemen didalam package  perpustakaan  pustaka  anggota | |
| Elemen diluar package  perpustakaan  metode  atribut  anggota  pustaka  atribut  metode | |

1. *Use case Diagram*

*Use case* atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antar satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, usecase digunakan untuk mengeahui fungsi apasaja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi fungsi itu.

Syarat penamaan pada usecase adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada usecase yaitu pendefinisian apa yang disebut aktor dan usecase.

1. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri.
2. Usecase merupakan fungsionalitas yang disediakan sistem sebagai unit unit yang saling bertukar pesan antar unti atau aktor.

Berikut adalah simbol yang ada pada diagram usecase :

Tabel 2.10 simbol pada diagram use case (AS Rosa dan M. Shalahuddin, 2013.156)

|  |  |  |
| --- | --- | --- |
| **No.** | **Simbol** | **Keterangan** |
| 1 | Aktor | Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri. |
| 2. | *Use Case* | Fungsionalitas yang disediakan sistem sebagai unit unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja diawal frase nama usecase |
| 3. | *Generelation* | Hubungan generalisasi dan spesialisasi (umum-khusus) antar dua buah usecase dimana fungsi yang satu adalah fungsi yang lebih umu dari lainnya. Arah panah mengarah pada usecase yang menjadi generalisasinya |
| 4. | *Extends* | Relasi usecase tambahan kesebuah usecase dimana usecase yang ditambahkan dapat berdiri sendiri walau tanpa usecase tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; |
| 5. | *Include* | Relasi usecase tambahan kedalams ebuah usecase dimana usecase yang ditambahkan memrlukan usecase ini untuk menjalankan fungsinya atau sebagai syarat dijalankan usecase ini |

1. *Activity Diagram*

Diagram aktivitas menggambarkan workflow atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak, yang perlu diperhatikan disini adlaah bahwa diagram aktivitas menggambarkan aktifitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal berikut :

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan
2. Urutan atau pengelompokan tampilan dari sistem / interface dimana setiap aktifitas dianggap memiliki sebuah rancangan antarmuka tampilan
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya
4. Rancangan menu yang ditampilkan pada perangkat lunak

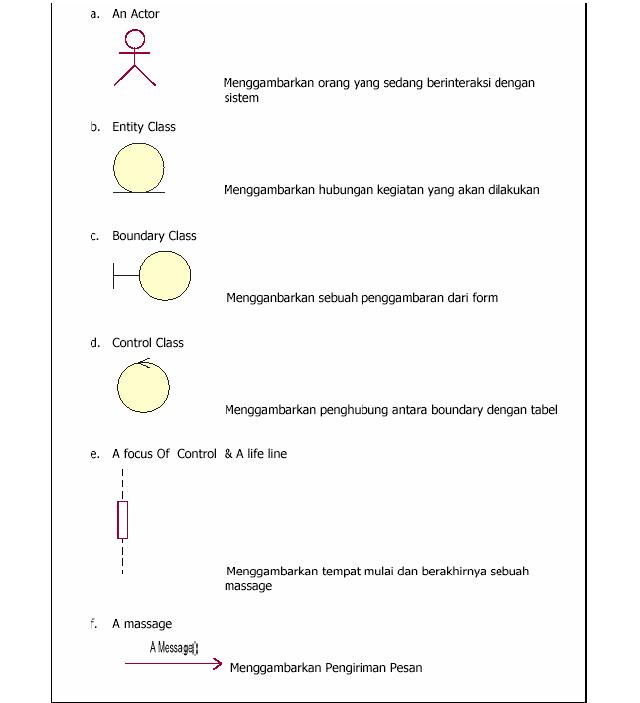
Berikut adalah simbol yang ada pada diagram aktivitas :

Tabel 2.11 simbol diagram aktivitas (AS Rosa dan M. Shalahuddin, 2013.162)

|  |  |  |
| --- | --- | --- |
| **No.** | **Simbol** | **Keterangan** |
| 1 | *Start State* | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal |
| 2. | *End State* | Status akhir yang diperlukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir |
| 3. | *State/Activities* | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja |
| 4. | Swimlane   |  | | --- | | Nama swimlane | |  | | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi |
| 6. | *Decision* | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu |

1. *Sequence diagram*

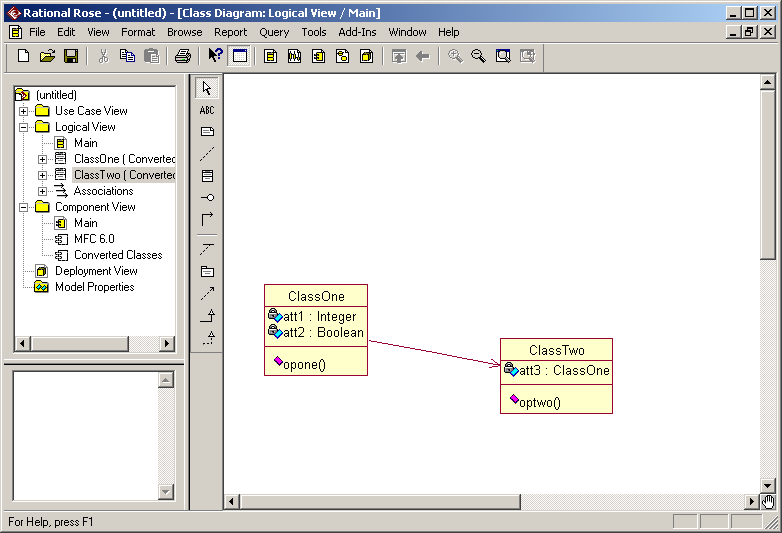
Diagram sekuen menggambarkan kelakuan objek pada usecase dengan medeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Banyaknya diagram sekuen yang harus digambarkan adalah minimal sebanyak pendefinisian usecase yang memiliki proses sendiri atau yang pening semua usecase telah didefinisikan . berikut adalah simbol yang ada pada diagram sekuen :



Gambar 2.13 simbol diagram sekuen

* 1. **Rational Rose**

Menurut **Adi Nugroho** (2005:20-21) dalam bukunya yang berjudul *Rational Rose* untuk Pemodelan Berorientasi Objek, menerangkan bahwza *Rational Rose* adalah perkakas (*tool*) pemodelan visual untuk pengembagan sistem berbasis objek yang sangat handal untuk digunakan sebagai bantuan bagi para pengembang dalam melakukan analisis dan perancangan sistem. *Rational Rose* digunakan untuk melakukan pemodelan sistem sebelum pengembangan menulis kode-kode dalam bahasa pemrograman tertentu.



Gambar2.14Antarmuka Awal Rational Rose

Dalam *Rational Rose*, pemodelan adalah cara melihat sistem dari berbagai sudut-pandang. Ia mencakup semua diagram yang dikenal dalam UML, aktor-aktor yang terlibat dalam sistem, use case, objek-objek, kelas-kelas, komponen-komponen, serta simpul-simpul penyebaran (*deployment node*). Selain itu, *Rational Rose* juga memungkinkan. Selain hal-hal di atas, *Rational Rose* juga akan membantu para pengembang dengan menghasilkan kode-kode inti dalam beberapa bahasa pemrograman.

* 1. **Teknik Pengujian Perangkat Lunak**
     1. **Pengujian *Black-Box***

Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program.

Pengujian black box berusaha menemukan kesalahan dalam kategori sebagai berikut: (1) fungsi-fungsi yang tidak benar atau hilang, (2) kesalahan interface, (3) kesalahan dalam struktur data atau akses databae eksternal, (4) kesalahan kinerja, (5) inisialisasi dan kesalahan terminasi (**Roger S. Pressman, Ph.D, 2002**).

Klasifikasi Black box testing mencakup beberapa pengujian, yaitu (**Janner Simarmata, 2010**) :

1. Pengujian fungsional (*functional testing*). Pada jenis pengujian ini, perangkat lunak diuji untuk persyaratan fungsional.
2. Pengujian tegangan (*stress testing*). Pengujian ini berkaitan dengan kualitas aplikasi di dalam lingkungan. Ide nya adalah untuk menciptakan sebuah lingkungan yang lebih menuntut aplikasi, tidak seperti saat aplikasi dijalankan pada beban kerja normal.
3. Pengujian beban (*load testing*). Pada pengujian ini, aplikasi akan diuji dengan beban berat atau masukan, seperti yang terjadi pada pengujian situs web, untuk mengetahui apakah aplikasi/situs gagal atau kinerjanya menurun.
4. Pengujian khusus (*ad hoc testing*). Jenis pengujian ini dilakukan tanpa penciptaan rencana pengujian (test plan) atau kasus pengujian (test case). Pengujian khusus membantu dalam menentukan lingkup dan durasi dari berbaga pengujian lainnya dan juga membantu para penguji dalam mempelajari aplikasi sebelum memulai pengujian dengan pengujian lainnya.
5. Pengujian penyelidikan (*exploratory testing*). Pengujian penyelidikan mirip dengan pengujian khusus dan dilakukan untuk mempelajari atau mencari aplikasi. Pengujian ini merupakan pendekatan yang menyenangkan untuk pengujian.
6. Pengujian usabilitas (*usability testing*). Pengujian ini dilakukan jika antarmuka pengguna dari aplikasinya penting dan harus spesifik untuk jenis pengguna tertentu.
7. Pengujian asap (*smoke testing*). Pengujian ini dilakukan untuk memeriksa apakah aplikasi tersebut sudah siap untuk pengujian yang lebih besar dan bekerja dengan baik tanpa cela sampai tingkay ang paling diharapkan.
8. Pengujian pemulihan (*recovery testing*). Pengujian ini pada dasarnya dilakukan untuk memeriksa seberapa cepat dan baiknya aplikasi bias pulih terhadap semua jenis crash atau kegagalan hardware, masalah bencana, dan lain-lain.
9. Pengujian volume (*volume testing*). Pengujian ini dilakukan terhadap efisiensi dari aplikasi. Jumlah data yang besar diproses melalui aplikasi (yang sedang diuji) untuk memeriksa keterbatasan ekstrem dari sistem.
10. Pengujian domain (*domain testing*). Pengujian ini merupakan penjelasan yang paling sering menjelaskan teknik pengujian.
11. Pengujian scenario (*scenario testing*). Pengujian ini adalah pengujian yang realistis, kredibel dan memotivasi stakeholder, tantangan untuk program dan mempermudah penguji untuk melakukan evaluasi.
12. Pengujian regresi (*regression testing*). Pengujian ini adalah gaya pengujian yang berfokus pada pengujian ulang (*retesting*) setelah ada perubahan.
13. Penerimaan pengguna (*user acceptance*). Pada pengujian ini, perangkat lunak akan diserahkan kepada pengguna untuk mengetahui apakah perangkat lunak memenuhi harapan pengguna dan bekerja seperti yang diharapkan.
14. Pengujian alfa (*alpha testing*). Pada pengujian ini pengguna akan diundang ke pusat pengembangan. Pengguna akan menggunakan aplikasi dan pengembangan mencatat setiap masukan atau tindakan yang dilakukan oleh pengguna.
15. Pengujian beta (*beta testing*). Pada pengujian ini, perangkat lunak didistribusikan sebagi sebuah versi beta dengan pengguna yang menguji aplikasi di situs mereka.
    * 1. **Pengujian *White-Box***

Pengujian *white box*, yang kadang disebut pengujian *glass box*, adalah metode desain *test* *case* yang menggunakan struktur control desain *procedural* untuk memperoleh *test case.* Dengan menggunakan metode pengujian *white box* perekayasa sistem dapat melakukan *test case* yang: (1) memberikan jaminan bahwa semua jalur independen pada suatu modul yang telah digunakan paling tidak satu kali. (2) menggunakan semua keputusan logis pada sisi *true* dan *false*. (3) mengeksekusi semua *loop* pada batasan mereka dan pada batas operasional mereka. (4) menggunakan struktur data internal untuk menjamin validitasnya (**Roger S. Pressman, Ph.D, 2002**).

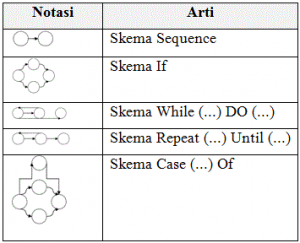
Klasifikasi white *box* testing mencakup beberapa pengujian, yaitu (**Janner Simarmata, 2010**) :

1. Pengujian unit. Pengujian ini berada pada tingkat yang sangat dasar seperti ketika unit kode dikembangkan atau fungsi tertentu dibangun. Pengujian unit berkaitan dengan unit secara keseluruhan.
2. Analisis statis dan dinamis. Analisis statis dilibatkan melalui kode untuk mengetahui segala kemungkinan cacat dalam kode, sedangkan analisis dinamis akan melibatkan pelaksanaan kode dan penganalisisan hasilnya.
3. Cakupan pernyataan. Dalam hal ini jenis pengujian kode dijalankan dengan setiap pernyataan dari aplikasi yang dijalankan minimal sekali. Hal tersebut membantu dalam memastikan semua pernyataan untuk dijalankan tanpa efek samping.
4. Cakupan cabang. Pengujian ini membantu pemvalidasian semua cabang di dalam kode dan memastikan bahwa tidak ada yang mengarah ke percabangan perilaku abnormal dari aplikasi.
5. Pengujian mutasi. Pada pengujian ini aplikasi diuji untuk kode yang telah dimodifikasi setelah pemasangan bug/cacat tertentu. Hal ini juga membantu dalam menemukan kode dan strategi pengodean yang dapat membantu dalam mengembangkan fungsi secara efektif.

Teknik *white box* sebagai berikut :

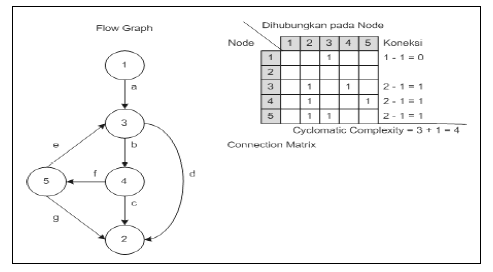
1. Pengujian basis *path*. Metode ini memungkinkan penguji dapat mengukur kompleksitas logis dari desain procedural dan menggunakannya sebagai pedoman untuk menetapkan himpunan basis dari semua jalur eksekusi.
2. Notasi Diagram Alir. Notasi yang digunakan untuk menggambarkan jalur eksekusi adalah notasi diagram alir (atau grafik program), yang menggunakan notasi lingkaran (simpul atau node) dan anak panah (link atau edge).

Tabel 2.12 Notasi Diagram Alir



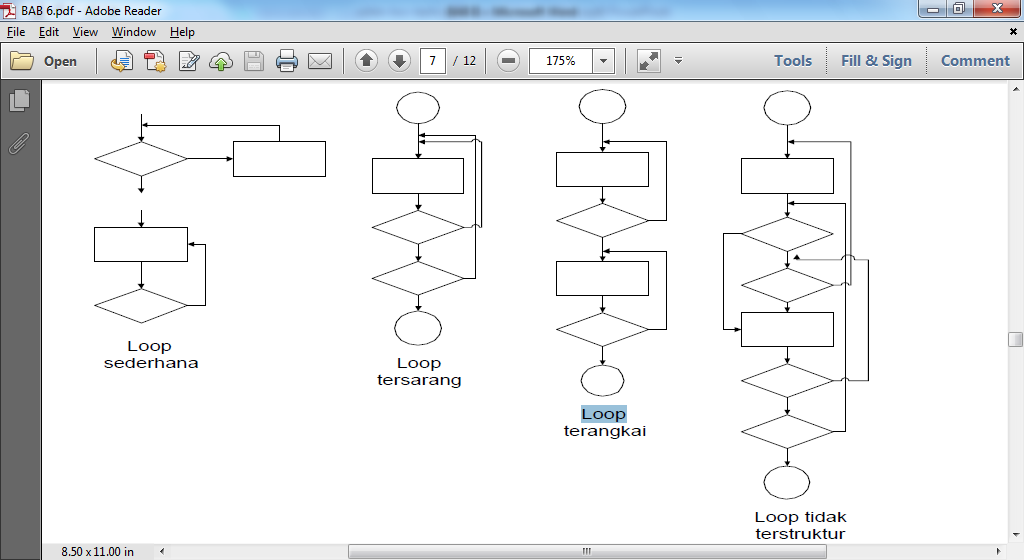
Gambar 2.15 Notasi Diagram Alir (Janner Simarmata, 2010)

1. Kompleksitas Siklomatis adalah metrics perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program, nilai yang didapat akan menentukan jumlah jalur independen dalam himpunan path, serta akan memberi nilai batas atas bagi jumlah pengujian yang harus dilakukan, untuk memastikan bahwa semua pernyataan telah dieksekusi sedikitnya satu kali.
2. Matriks Grafis (Graph Matrik). Bentuk struktur data yang sering digunakan untuk mengambarkan pengujian adalah dengan matriks grafis. Matriks grafis adalah matriks bujursangkar yang berukuran sama dengan jumlah simpul pada grafik alir. Inputan dalam matriks harus bersesuaian dengan arah sisi dengan simpul.



Gambar 2.16 Matriks Grafis (Janner Simarmata, 2010)

1. Pengujian Struktur Kontrol. Teknik pengujian basis path merupakan salah satu dari sejumlah teknik untuk pengujian struktur control, namun pengujian basis path tidak memadai untuk beberapa kasus uji.
2. Pengujian Kondisi. Pengujian kondisi menggunakan kondisi logis sederhana yang terdapat dalam program. Bila suatu kondisi tidak benar, maka akan terdapat paling tidak satu komponen dari kondisi yang salah, sehingga tipe kesalahan pada suatu kondisi meliputi :
3. Kesalahan operator Boolean
4. Kesalahan variable Boolean
5. Kesalahan tanda kurung Boolean
6. Kesalahan operator relasional
7. Kesalahan persamaan aritmatika
8. Pengujian Aliran Data. Metode pengujian aliran data melakukan pengujian dengan mengggunakan definisi variable dalam program, efektif digunakan untuk melindungi kesalahan, tetapi akan memiliki cakupan pengukuran dan pemilihan jalur uji yang kompleks.
9. Pengujian *Loop*. *Loop* atau skema pengulangan sering digunakan dalam pembuatan program. Terdapat beberapa macam *loop*, yaitu:
10. *Loop* Sederhana. Pengujian yang dilakukan harus memperhatikan hal-hal yaitu mengabaikan keseluruhan *loop*, hanya terdapat satu jalur yang melewati *loop*, suatu variable akan melewati loop jika bernilai lebih besar dari nilai yang ditentukan, harus memperhatikan batasan variable pada loop. (kondisi : n-1, n, n+1)
11. *Loop* Tersarang. Jumlah pengujian akan bertambah secara geometris sesuai jumlah persarangan *loop* yang ada. Untuk menyederhanakan pengujian, maka harus diperhatikan hal-hal berikut yaitu memulai pengujian dari loop terdalam, dengan memulai pengujian dari nilai minimum, melakukan pengujian loop sederhana dari loop terdalam hingga loop terluar dengan memperhatikan parameter yang digunakan.
12. Loop Terangkai. Pengujian untuk loop terangkai harus disesuaikan dengan independensi variable antara loop tersebut. Jika variable yang digunakan dalam loop kedua tidak bergantung dengan loop pertama, maka digunakan pengujian loop sederhana, sedangkan bila loop kedua bergantung secara nilai dengan loop kedua, maka lakukan pengujian tersarang.
13. Loop Tidak Terstruktur. Sama sekali tidak dianjurkan untuk digunakan dalam membuat program.



Gambar 2.17. pengujian Loop (Janner Simarmata, 2010)