BAB 2

LANDASAN TEORI

2.1 Rekayasa Perangkat Lunak

2.1.1 Pengertian Perangkat Lunak

Menurut Pressman (2012,p5) perangkat lunak adalah instruksi-instruksi(program komputer) yang ketika dijalankan menyediakan fitur-fitur, fungsi-fungsi,dan kinerja-kinerja yang dikehendaki; Struktur data yang memungkinkan program-program memanipulasi informasi; Dan informasi deskriptif pada salinan tercetak dan bentuk-bentuk maya yang menggambarkan pengoperasian dan penggunaan program-program.

Mengacu pada pendapat Fritz Bauer dalam Pressman (2012,p15), perangkat lunak adalah pembuatan dan penggunaan prinsip-prinsip penting rekayasa supaya pengguna bisa memperoleh perangkat lunak secara murah yang dapat diandalkan dan bekerja secara efisien pada mesin-mesin yang sesungguh.

Berdasarkan beberapa pendapat diatas, perangkat lunak dapat disimpulkan sebagai instruksi pengunaan program komputer yang ditampilkan memungkinkan untuk pengguna memanipulasi informasi, yang dapat di peroleh pengguna secara murah dan dapat bekerja secara efisisen.

2.1.2 Karakteristik-karakteristik perangkat lunak

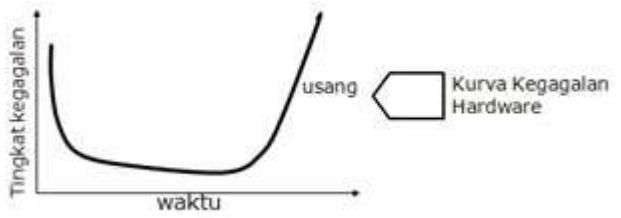
Perangkat lunak memiliki tiga karakteristik utama yaitu sebagai berikut:

1. Perangkat lunak dikembangkan atau direkayasa

   Kualitas tinggi dicapai melalui perancangan yang bagus, namun fase produksi pada suatu produksi pada suatu produk perangkat keras dapat menunjukkan masalah kualitas yang lebih kelihatan(atau lebih mudah dikoreksi) dibandingkan pada apa yang ada pada suatu produk perangkat lunak.

   Kedua kegiatan bergantung pada manusia, namun hubungan antara manusia dengan apa yang dihasilakan benar-benar berbeda. Kedua kegiatan membutuhkan konstruksi suatu ”produk”, tetapi pendekatan-pendekatannya cukup berbeda.

2. Perangkat lunak tidak mengalami “kelemahan”

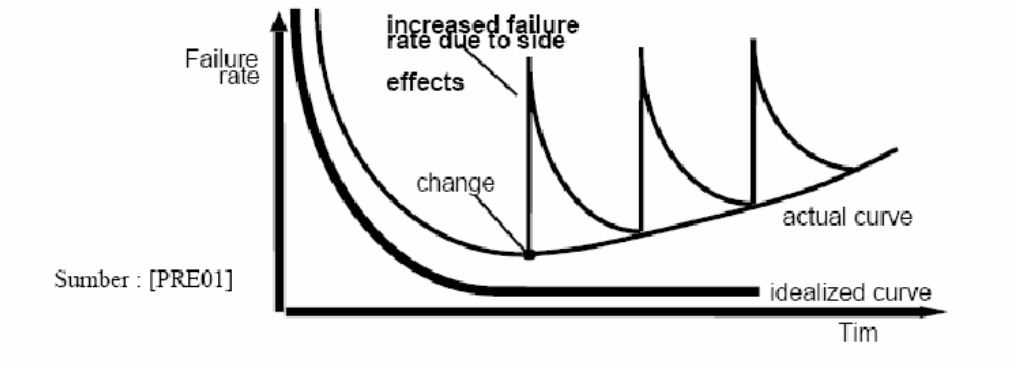


Gambar 2.1 Karakteristik Perangkat Lunak

Menggambarkan angka kegagalan berbanding dengan fungsi waktu pada perangkat keras. Hubungan tersebut, sering disebut sebagai “*kurva bathtub*” menunjukkan bahwa perangkat keras menunjukkan angka kegagalan relatif tinggi pada awal kehidupannya.

Perangkat lunak pada dasarnya tidak terpengaruh oleh kondisi-kondisi lingkungan yang menyebabkan perangkat keras mengalami kelelahan.

Oleh karena itu, secara teoritis kurva angka kegagalan pada perangkat lunak harus membentuk “kurva ideal” yang ditunjukkan pada gambar:



Gambar 2.2 Karakteristik Perangkat Lunak

Cacat yang tidak ditemukan akan menyebabkan angka kegagalan tinggi pada awalnya dalam kehidupan suatu program komputer.

Namun, angka-angka tersebut terkoreksi dan kurva mendatar seperti yang ditunjukkan. Kurva ideal merupakan penyederhanaan kasar dari model kegagalan yang actual pada perangkat lunak.

3. Meskipun insdustri beralih ke konstruksi berbasis komponen, sebagian besar perangkat lunak masih tetap dibuat berdasarkan spesifikasi yang diminta pengguna.

   Sebuah komponen perangkat lunak harus dirancang dan diterapkan sedemikian rupa sehingga dapat digunakan kembali dalam program yang berbeda-beda. Komponen yang dapat digunakan ulang(*reusable*) yang modern dapat membungkus di dalamnya data dan pemrosesan data itu sendiri. Hal tersebut memungkinkan rekayasawan-rekayasawan perangkat lunak untuk membuat aplikasi-

aplikasi baru dari bagian-bagiannya yang bersifat dapat digunakan ulang.

2.2 Sistem Basis Data

2.2.1 Pengertian Basis Data

Menurut Connolly dan Begg (2005, p15) *database* merupakan sebuah koleksi berbagi data logis yang terkait dan deskripsi dari data ini sendiri yang dirancang untuk memenuhi kebutuhan informasi dari suatu organisasi.

2.2.2 Sistem Manajemen Basis Data

Menurut Connolly dan Begg (2005, p16) sistem manajemen basis data adalah sebuah sistem perangkat lunak yang memungkinkan pengguna untuk akses ke *database*.

Komponen-komponen yang terdapat dalam lingkungan sistem manajemen basis data adalah sebagai berikut (Connolly dan Begg, 2005, pp18-21).

1 Perangkat Keras (*Hardware*)

Sistem manajemen basis data dan aplikasi membutuhkan perangkat keras untuk dapat berjalan. Perangkat keras yang digunakan dapat berupa *personal computer*, *mainframe*, hingga sebuah jaringan komputer, tergantung pada kebutuhan organisasi dan jensi sistem manajemen basis data yang digunakan. Sebuah DBMS membutuhkan jumlah minimal

memori utama dan ruang disk untuk menjalankan, tetapi ini konfigurasi minimal mungkin tidak selalu memberikan kinerja yang dapat diterima.

2. Perangkat Lunak (*software*)

Komponen perangkat lunak dalam lingkungan sistem manajemen basis data terdiri dari sistem manajemen basis data itu sendiri, program aplikasi, beserta sistem operasi, termasuk juga perangkat lunak untuk jaringan jika sistem manajemen basis data tersebut digunakan dalam sebuah jaringan.

3. Data

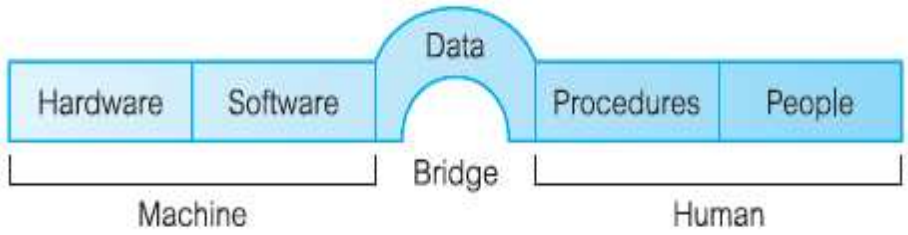
Data adalah komponen yang paling penting dari lingkungan DBMS. Dalam lingkungan sistem manajemen basis data, data merupakan komponen yang berperan sebagau jembatan antara komponen mesin yaitu perangkat keras dan perangkat lunak, dengan komponen manusia yaitu prosedur dan manusia itu sendiri. Sebuah basis data memiliki data operasional dan *meta-data* yaitu data mengenai data.

4. Prosedur

Merupakan kumpulan instruksi dan aturan yang menjaga desain dan penggunaan sebuah basis data. Pengguna sistem dan pengelola basis data membutuhkan sebuah prosedur mengenai cara penggunaan sistem manajemen basis data.

5. Manusia

Komponen terakhir dari lingkungan sistem manajemen basis data yang melakukan akses ke baris data, baik berupa instruksi modifikasi ataupun pengelolaan sumber daya basis data.



Gambar 2.3 Lingkungan sistem manajemen basis data (Connolly dan Begg, 2005, p19)

2.2.3 Pengertian MySql

Menurut Welling dan Thomson (2001,p3), MySql (diucapkan-My Ess-Que-Ell) adalah *Relational Database Management System* (RDBMS) yang sangat cepat, dan kuat. Sebuah database memungkinkan pengguna untuk secara efisien menyimpan, mencari, mengurutkan, dan mengambil data. Server MySql mengontrol akses ke data pengguna untuk memastikan bahwa beberapa pengguna dapat bekerja secara bersamaan, untuk menyediakan akses yang cepat, dan memastikan bahwa hanya pengguna yang berwenang dapat memperoleh akses. Oleh karena itu, MySql adalah *multi-user, multi-threaded server*. SQL (Structured Query Language), dijadikan bahasa *database* standar di seluruh dunia.

2.3 *Internet*

Internet adalah dunia internetwork terbesar, dimana suatu informasi dapat menyebar lebih ke jutaan komputer orang diluar sana yang dapat terhubung ke jutaan orang lainnya, baik itu di rumah-rumah, bisnis, sekolah, dan kantor pemerintah di seluruh dunia, jutaan komputer dari semua jenis-PC, Macintoshes, *mainframe* perusahaan besar, dan lain-lain-yang terhubung bersama-sama dalam jaringan, dan jaringan-jaringan yang terhubung ke satu sama lain yang berbeda untuk membentuk Internet. Karena segala sesuatu terhubung, setiap komputer di Internet dapat berkomunikasi dengan komputer lain di Internet. Ketika komputer terhubung bersama-sama dalam sebuah jaringan, pengguna komputer tersebut dapat saling mengirim pesan lain dan berbagi *file* komputer dan program.

Jaringan komputer terkecil saat ini adalah ketika dua PC terhubung bersama-sama di kantor. Mereka dapat menjadi besar seperti ribuan komputer dari semua jenis tersebar di seluruh dunia dan terhubung satu sama lain bukan hanya dengan kabel, tetapi melalui saluran telepon dan bahkan melalui udara melalui satelit. (Snell et.al, 2003,p1).

2.4 *Web Application*

Menurut Mercer (2005,p392), aplikasi web adalah aplikasi yang diakses melalui definisi web. ini berarti pengguna/*user* yang akan mengakses aplikasi menggunakan beberapa jenis *browser* sebagai *client*. Meskipun

demikian, dalam banyak skenario logika aplikasi dan data pendukung aplikasi berada pada server yang sama.

2.4.1 web arsitektur

Menurut Mattison (1999,p357), Pada dasarnya, *World Wide Web* adalah layanan manajemen layar yang berjalan di dalam internet. Dua komponen utama dari sesi Web adalah klien, suatu produk browser internet seperti Netscape Navigator atau MS Internet Explorer, dan server, sebuah produk Web server seperti server perusahaan Netscape atau server Microsoft.

2.4.2 Pengertian PHP

Menurut Welling dan Thomson (2001,p2), PHP adalah bahasa *server-side scripting* yang dirancang khusus untuk web. Dalam sebuah halaman HTML, User dapat menanamkan kode PHP yang akan di eksekusi setiap kali halaman di kunjungi. Kode PHP user diinterpretasikan pada server web dan menghasilkan output HTML atau lainnya yang akan di lihat pengunjung web.

2.4.2.1 Kelebihan PHP

Menurut Welling dan Thomson (2001,p4) Beberapa pesaing PHP adalah Perl, Microsoft Active Server Pages(ASP), Java Server Pages(JSP), dan Allaire Cold Fusion.

Dibandingkan dengan produk tersebut, PHP memiliki Kelebihan, termasuk yang sebagai berikut:

1. Kinerja tinggi,
2. Antarmuka untuk banyak sistem database yang berbeda,
3. *Built-in* perpustakaan untuk banyak tugas web umum,
4. Biaya rendah,
5. Kemudahan belajar dan penggunaan,
6. Portabilitas,
7. Ketersediaan kode sumber.

2.5 Interaksi Manusia dan Komputer

2.5.1 Pengertian Interaksi Manusia dan Komputer

Menurut Shneiderman dan Plaisant (2010,pp4-5), interaksi manusia dan komputer adalah tampilan antarmuka yang dapat digunakan pengguna untuk berinteraksi dengan komputer.

2.5.2 Prinsip Perancangan Antarmuka

Prinsip delapan aturan emas untuk merancang antarmuka menurut Shneiderman dan Plaisant (2010, pp74-75) yaitu sebagai berikut :

1. Konsistensi

   Desain antarmuka yang baik harus memiliki konsistensi pada setiap aksi yang dilakukan, seta pada setiap menu, warna, jenis huruf, dan unsur-unsur lain.

2. Memenuhi kebutuhan pengguna secara universal

Perancangan fitur serta penggunaan simbol harus dapat dimengerti oleh semua kalangan pengguna. Seperti menyediakan fitur untuk pengguna awam seperti tutorial, dan fitur untuk pengguna mahir seperti *shortcut*.

3. Memberi umpan balik yang informatif

Pada setiap aksi yang dilakukan pengguna, harus ada respon yang dapat mengarahkan pengguna untuk lebih memahami instruksi yang sedang dilakukan. Respon tersebut dapat berupa umpan balik dari sistem terhadap input yang dimasukkan oleh pengguna.

4. Perancangan dialog yang menghasilkan keadaan akhir

Sistem harus memiliki urutan aksi yang jelas yaitu memiliki keadaan awal, pertengahan dan akhir sistem.

5. Pencegahan kesalahan

Menjaga pengguna dari melakukan kesalahan. Selain itu, sistem harus memberikan informasi mengenai kesalahan yang terjadi.

6. Pembalikan aksi yang mudah

Sistem harus dapat memberikan pembalikan aksi dari situasi yang sedang dilakukan ke situasi sebelumnya.

7. Pusat Kendali internal

Pengguna dapat memiliki kendali atas antarmuka sehingga pengguna merasa mendapatkan kemudahan dalam menggunakan antarmuka.

8. Mengurangi beban jangka pendek

Sistem harus memiliki urutan aksi yang jelas dan mudah diingat oleh pengguna untuk mengurangi beban ingatan jangka pendek.

2.6 *Unified Modeling Language* (UML)

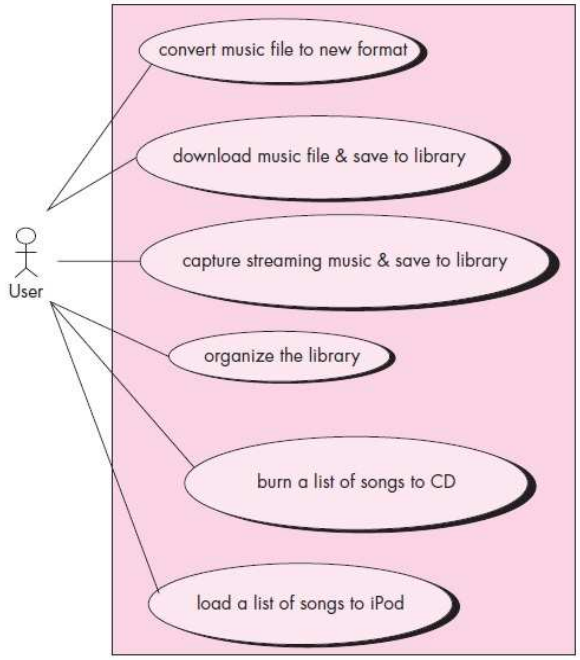
Unified Modeling Language adalah bahasa standar untuk menulis rancangan *software*. UML dapat digunakan untuk membangun dan mendokumentasikan kerangka *software*. Di sisi lain, dengan membangun rancangan arsitektur untuk digunakan para konstruksi perusahaan, pembangun *software* membuat UML diagram untuk membantu pengembang *software* untuk membangun *software*. Sehingga lebih mudah untuk dipahami, mudah untuk menentukan sistem dan mudah untuk menjelaskan desain system (Pressman, 2010, p841).

2.6.1 *Use Case Diagram*

Model *use case diagram* membantu untuk menentukan fungsi dan fitur *software* dari perspektif pengguna. *Use case* menjelaskan bagaimana pengguna berinteraksi dengan sistem dengan cara mendefinisikan langkah-langkah yang diperlukan untuk mencapai tujuan tertentu. *Use case diagram* merupakan gambaran dari semua *use case* dan hubungan antar setiap *use case.*

Dalam *use case diagram, use case* ditampilkan dalam bentuk oval. Para aktor terhubung dengan setiap *use case* menggunakan garis. Karena *use case diagram* menampilkan semua *use case,* hal ini sangat membantu untuk

memastikan bahwa semua fungsi dari sistem sudah tercakup (Pressman, 2010, pp847-848).



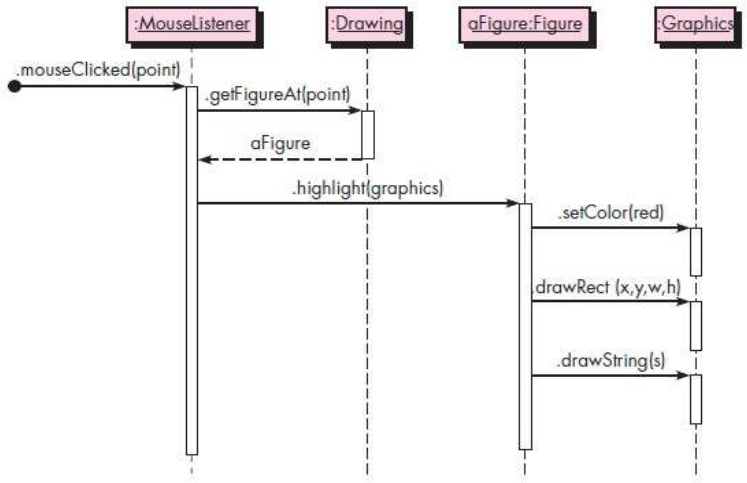
Gambar 2.4 *Contoh Use case diagram* (Pressman, 2010, p848).

2.6.2 *Sequence Diagram*

Berbeda dengan *class diagram* yang menampilkan struktur statis dari komponen sistem *software*, *sequence diagram* digunakan untuk menampilkan struktur yang dinamis antara obyek selama fungsi dijalankan. *Sequence diagram* menampilkan proses pengiriman pesan antar obyek untuk menyelesaikan fungsi tertentu. Alasan lain dengan menggunakan *sequence diagram* adalah untuk menampilkan interaksi suatu *use case* atau satu skenario sistem *software.*

Sequence diagram menampilkan metode panggilan menggunakan panah horizontal dari pelaku ke target pelaku, diberi label dengan nama metode dan

termasuk parameter, jenis dan jenis timbal balik. Untuk kasus *looping*, *conditional*, dan struktur kontrol lainnya dalam *sequence diagram*, dapat menggunakan *frame* interaksi seperti persegi panjang yang mengelilingi bagian dari diagram (Pressman, 2010, pp848=850).



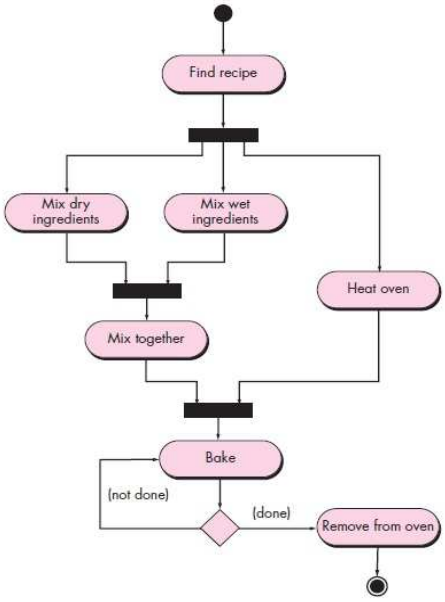
Gambar 2.5 *Contoh sequence diagram* (Pressman, 2010, pp850).

2.6.3 *Activity Diagram*

UML model *activity diagram* menjelaskan perilaku dinamis dari sistem atau bagian bagian sistem melalui aliran proses yang dilakukan sistem. Hal ini sama dengan model *flowchart* tetapi sedikit berbeda dikarenakan *activity diagram* dapat menampilkan aliran proses sistem secara bersamaan.

Komponen utama dari *activity diagram* adalah *action node*, diwakili oleh bulat persegi panjang, yang sesuai dengan tugas yang dilakukan oleh *software*. Panah dari satu *node* ke *node* lain menjelaskan aliran kontrol. Hal ini berarti bahwa setelah *action* pertama selesai, *action* kedua baru dijalankan. Sebuah lingkaran hitam pekat menjelaskan awal proses *activity* dimulai. Sebuah titik hitam yang yang dikelilingi lingkaran hitam menjelaskan akhir dari proses

activity. Garis horizontal berwarna hitam merupakan pemisah dua *action* atau lebih secara bersamaan (Pressman, 2010, p853).



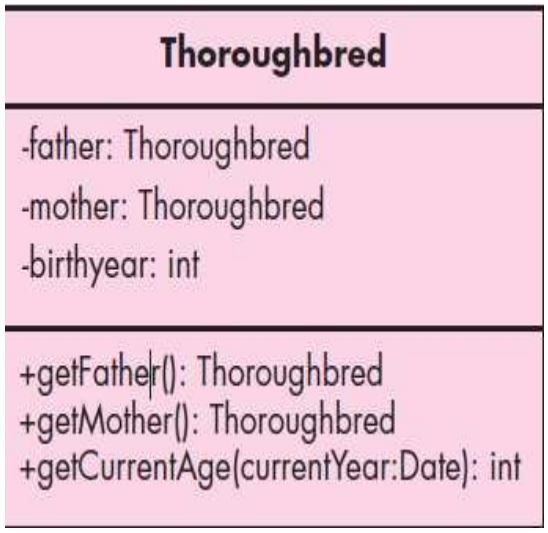
Gambar 2.6 *Contoh activity diagram* (Pressman, 2010, p854).

2.6.4 *Class Diagram*

Untuk kelas model, termasuk atribut kelas, pengoperasian, dan hubungan antar kelas dengan kelas lain, UML menyediakan model *class diagram*. *Class* diagram menyediakan *view* statis atau struktural dari sistem. *Class diagram* tidak menunjukkan sifat dinamis dari komunikasi antar objek setiap kelas dalam diagram.

Unsur utama dalam *class* diagram adalah kotak, yang merupakan ikon yang digunakan untuk mewakili kelas-kelas. Setiap kotak dibagi dengan garis horizontal. Bagian atas berisi nama *class*. Bagian tengah berisi atribut tiap kelas. Setiap atribut mengacu pada sesuatu yang merupakan objek pada kelas.

Setiap atribut dapat memiliki nama, tipe dan simbol. Simbol ditunjukkan dengan -, #, ~ atau +, yang masing masing menunjukan atribut tersebut *private, protected, package* atau *public* (Pressman, 2010, pp842-843).

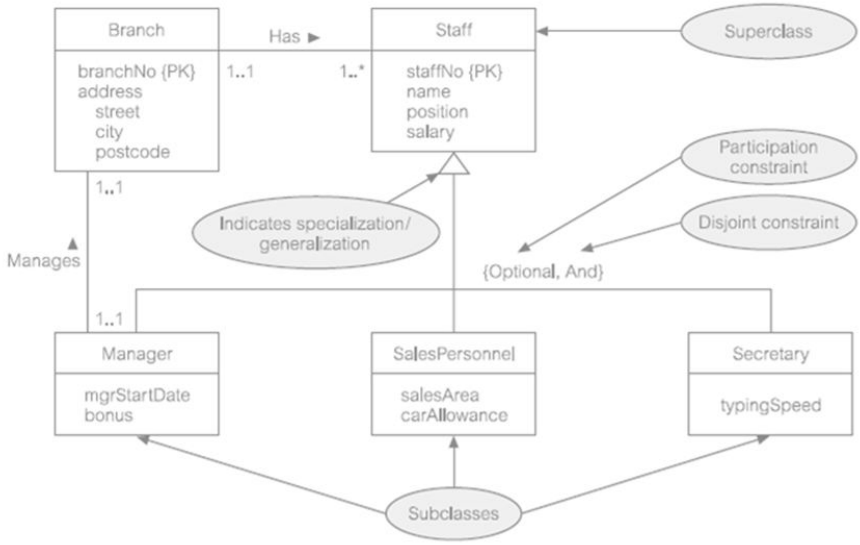


Gambar 2.7 *Contoh Class diagram* (Pressman, 2010, pp842-843).

2.6.5 *Entity Relationship Diagram (ERD)*

Menurut Connolly dan Begg (2005, p342) ERD adalah top-down pendekatan desain database yang dimulai dengan mengidentifikasi data penting yang disebut entitas dan hubungan antara data yang harus direpresentasikan dalam model. kemudian tambahkan lebih detail seperti informasi yang ingin diteruskan tentang entitas dan hubungan disebut atribut dan setiap kendala pada entitas, hubungan, dan atribut.

Gambar 2.8 *Contoh ERD* (Connolly dan Begg, 2005, p358).



Gambar 2.9 *Contoh ERD* (Connolly dan Begg, 2005, p376).

2.6.6 Model - View – Controller pattern

Menurut Sommerville (2011,p155) Model-View-Controller (MVC) adalah pola arsitektur yang memisahkan presentasi dan interaksi dari sistem. sistem ini terbagi dalam tiga komponen logis yang berinteraksi satu sama lain. Komponen *Model* mengelola sistem data dan operasi yang terkait pada data. Komponen *View* mendefinisikan dan mengelola interaksi pengguna. Komponen *Controller* mengelola interaksi pengguna (misalnya pada penekanan tombol, klik mouse, dll) dan melewati interaksi ini ke *View* dan *Model*. Dengan demikian,

desain MVC memberikan pemisahan tugas dan tanggung jawab yang jelas antara Komponen *Model*, *View*, dan *Controller*.

Tujuan penggunaan MVC dalam pemrograman adalah untuk mengelompokkan fungsi – fungsi yang ada dan berserakan menjadi sebuah kesatuan sesuai dengan tipenya masing – masing. Misalnya fungsi – fungsi yang digunakan untuk mengakses *database* disatukan dalam satu tempat, kemudian fungsi untuk menampilkan tampilan *website* dalam satu tempat yang lain.

Dengan terpisahnya antara fungsi logika program serta tampilan dari program, maka akan memudahkan proses *web development.* khususnya ketika kita harus bekerja dalam sebuah tim. Masing – masing anggota tim bisa mengerjakan pekerjaannya sendiri. *Programmer* bisa mengerjakan tugasnya yang berkaitan dengan logika program, sedangkan desainer bisa mengerjakan tugasnya yang berkaitan dengan proses penampilan *website* tersebut.

1. Model

Implementasi aplikasi data. Komponen ini merupakan kode utama yang melakukan kerja internal aplikasi. komponen ini merupakan kode utama yang melakukan kerja internal aplikasi. Komponen *Model* tidak tahu sama sekali bagaimana bentuk komponen *View* dan *Controller*.

2. View

Implementasi *layer* presentasi yang berinteraksi dengan pengguna. Komponen *View* memerlukan data dari *Controller* untuk meneruskan *input* pengguna ke komponen *Model*.

3. Controller

Bertindak sebagai pusat kendali aplikasi dan penanggungjawab lalu lintas data ke komponen *Model* dan *View*. Controller mengatur *Model* dan *View* dengan bereaksi terhadap data yang dikirimkan oleh pengguna.

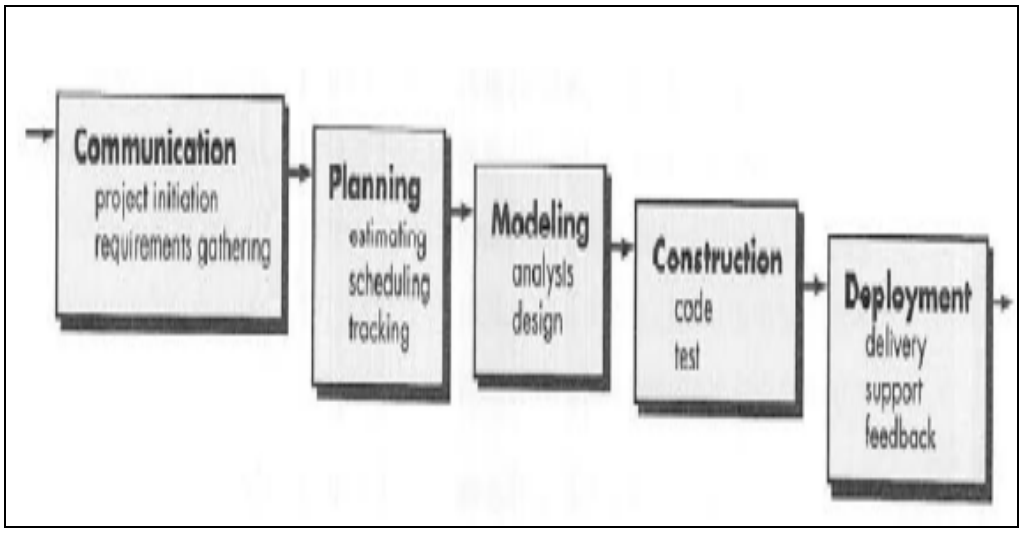
Proses MVC yang terjadi dalam suatu kasus *user request* adalah sebagai berikut :

1. Pengguna melalui browser mengirimkan suatu request untuk suatu halaman kepada *controller* yang berada di *server*.
2. *Controller* mengambil data yang diperlukan dari *model* dalam hal melakukan *respond* terhadap request.
3. *Controller* mengolah halaman dan mengirimkannya kepada *view.*
4. *View* mengirim halaman kembali melalui *browser* agar dapat dilihat oleh *client.*

2.7 *Waterfall Model*

Model air terjun *(Waterfall)* kadang dinamakan siklus hidup klasik (*Classic life cycle*), dimana hal ini menyiratkan pendekatan yang sistematis dan berurutan (sekuensial) pada pengembangan perangkat lunak, yang dimulai dengan spesifikasi kebutuhan pengguna dan berlanjut melalui tahapan – tahapan perencanaan (*planning*),

pemodelan (*modeling*), konstruksi(*construction*), serta penyerahan sistem/perangkat lunak ke para pelanggan/pengguna (*deployment*), yang diakhiri dengan dukungan berkelanjutan pada perangkat lunak lengkap yang dihasilkan (Pressman, 2012, p46).



Gambar 2.10 Contoh Gambar Urutan Proses dari *Waterfall Model*

(Pressman, 2012, p46)

Berikut adalah penjelasan dari tahapan-tahapan tersebut menurut Pressman (2012, p46):

1. *Communication*

   Sebelum pekerjaan teknis untuk pembuatan sistem dimulai, sebaiknya terlebih dahulu berkomunikasi dengan pelanggan dan *stakeholder* lainnya. Hal ini sangat penting untuk mengetahui sistem apa yang akan dibuat serta juga untuk membantu dalam mengumpulkan informasi terkait pengembangan atau pembuatan sistem.

2. *Planning*

   Tahap berikutnya adalah perencanaan (*planning*). Tahap ini berguna untuk bisa lebih mengatur kinerja para *software engineer*, mengetahui resiko apa saja yang akan dihadapi, mengetahui apa saja sumber daya yang dibutuhkan, apa yang akan dihasilkan, serta yang terpenting adalah mengatur jadwal pembuatan perangkat lunak.

3. *Modeling*

   Pada tahap ini, sebuah model akan dibuat untuk lebih memahami perangkat lunak atau sistem yang seperti yang akan dihasilkan. Serta dengan tahap pemodelan, *software engineer* juga dapat lebih mengetahui desain perangkat lunak atau sistem yang seperti apa agar dapat menyelesaikan masalah yang sedang berlangsung.

4. *Construction*

   Tahap ini menggabungkan antara membuat *code* untuk *coding* (bisa manual atau otomatis) dan *testing* yang dibutuhkan untuk mengetahui kesalahan-kesalahan apa saja yang terdapat pada sistem atau perangkat lunak.

5. *Deployment*

   Tahap yang terakhir adalah *Deployment*, dimana pada tahap ini perangkat lunak yang telah selesai akan dipasarkan ke pelanggan serta juga mengumpulkan kritik dan saran dari para pelanggan tersebut.

2.8 Pengertian Data Flow Diagram (DFD)

Menurut Pressman (2012, p225) DFD memperlihatkan gambaran tentang masukan-proses-keluaran dari suatu sistem/perangkat lunak. Yaitu, objek-objek data mengalir ke dalam perangkat lunak, kemudian objek-objek data itu akan di transformasi oleh elemen – elemen pemrosesan, dan objek – objek data hasilnya akan mengalir keluar dari sistem/perangkat lunak. Objek – objek data, dalam penggambaran DFD, biasanya direpresentasikan menggunakan tanda panah berlabel, dan transformasi – transformasi biasanya di representasikan menggunakan lingkaran – lingkaran. DFD pada dasarnya digambarkan dalam bentuk hierarki. Yaitu DFD yang pertama (sering disebut sebagai DFD peringkat 0 atau diagram konteks) menggambarkan sistem secara keseluruhan. DFD – DFD berikutnya sesungguhnya merupakan penghalusan dari diagram konteks, memberikan gambaran yang semakin rinci dari diagram konteks, dan hal ini akan berlanjut ke peringkat – peringkat selanjutnya.

DFD memiliki tiga komponen di dalamnya, yaitu:

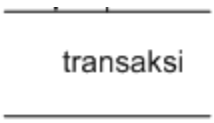
1. External Entity

Merupakan entitas luar yang berinteraksi langsung dengan sistem, bisa dalam bentuk memberikan data ke dalam sistem, menerima data dari sistem, atau keduanya.

2. Proses

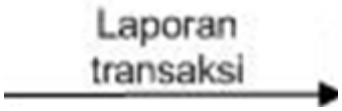
Merepresentasikan proses yang terjadi di dalam sistem, biasanya digambarkan dalam bentuk lingkaran(bubble).

3. Data store



Merepresentasikan tempat penyimpan data dalam sistem.

4. Aliran data



Merepresentasikan data yang mengalir dari sumber data ke tujuan, bisa dari proses ke proses lain, dari external entity ke proses, atau dari proses ke external entity.

Satu hal penting yang menjadi *concern* utama dalam pembuatan DFD adalah konsistensi. Berikut beberapa aturan untuk menjaga konsistensi DFD:

1. Proses

a. menggunakan label berupa kata kerja (misalkan: memproses gambar),

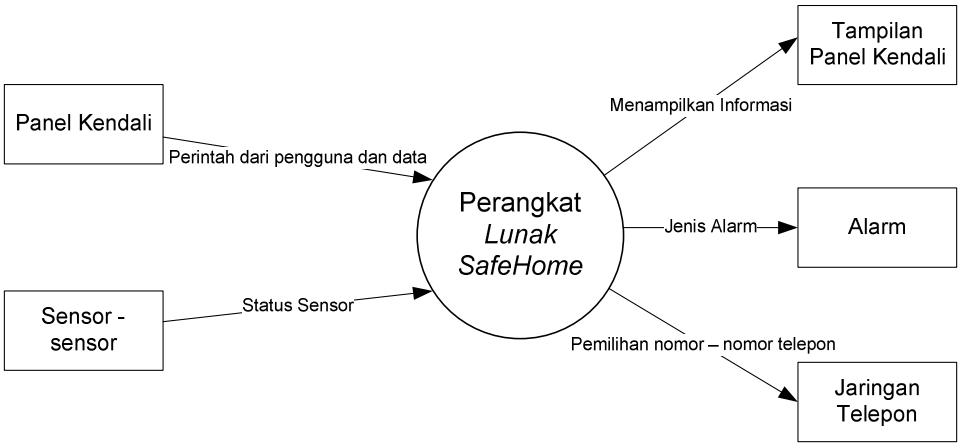
b. minimal memiliki satu masukan dan satu keluaran.

2. Data store

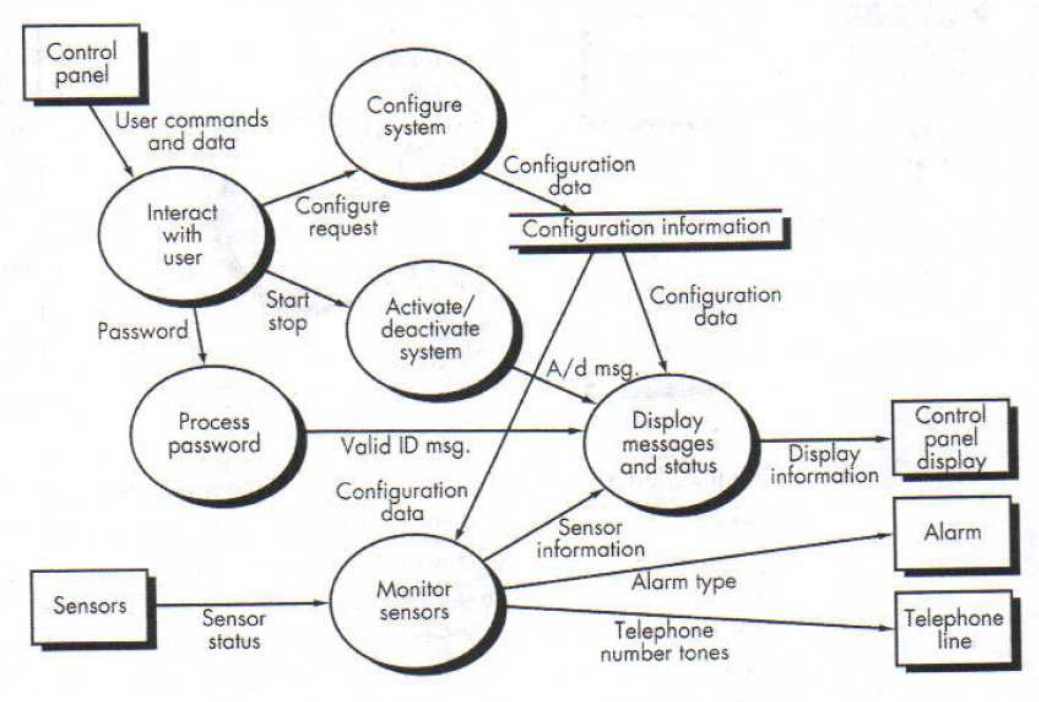
a. data tidak dapat dipindahkan secara langsung dari *data store* satu ke *data store* yang lain,
b. data tidak dapat dipindahkan langsung dari external entity ke data store, dan sebaliknya dari data store ke *external entity*,
c. menggunakan label berupa kata benda (misalkan: gambar).

3. Aliran data

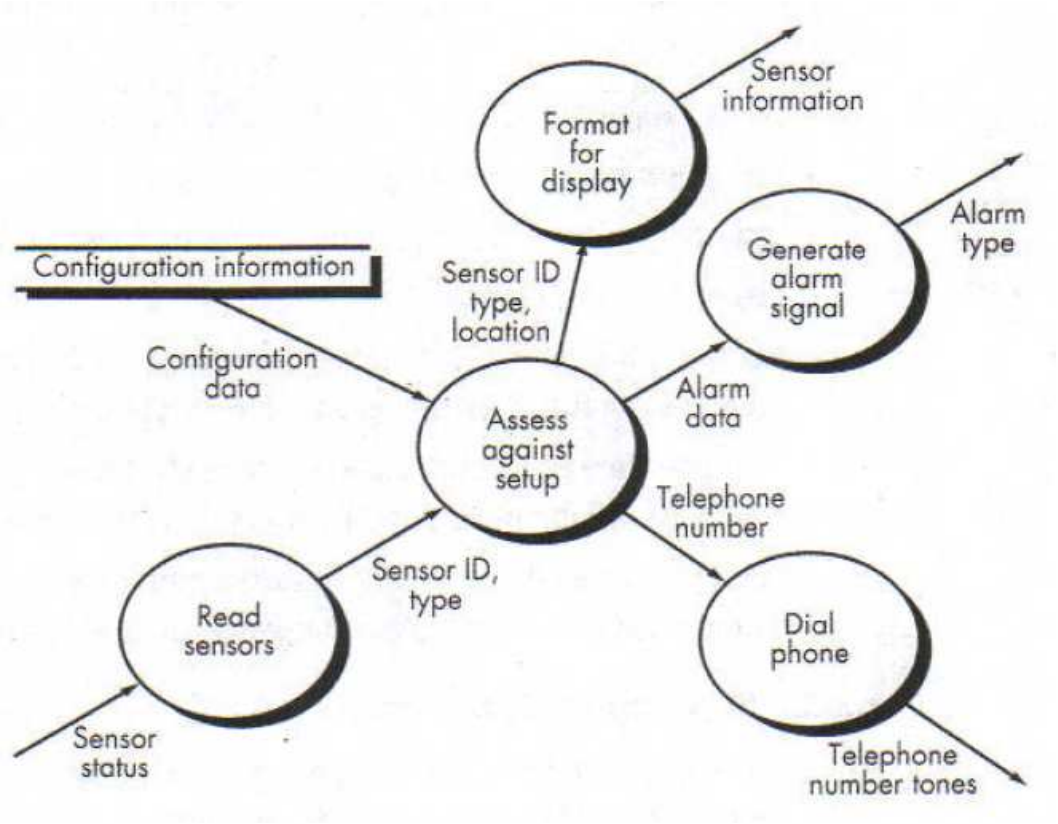
a. hanya memiliki satu arah,
b. data tidak dapat kembali secara langsung ke proses yang sama,
c. aliran data ke data store berarti melakukan pengubahan data(hapus atau ubah),
d. aliran data dari data store berarti mengambil data,
e. menggunakan label berupa kata benda.



Gambar 2.11 Contoh Diagram Konteks (Pressman 2012, p225)



Gambar 2.12 Contoh DFD peringkat 1 (Pressman 2012, p228).



Gambar 2.13 Contoh *DFD* peringkat 2 (Pressman 2012, p228).

2.9 Pengertian *State Transition Diagram*

Menurut Rajaraman dan Radhakrishnan (2004, p175) STD atau *State Transition Diagram* terdiri dari jumlah node. Setiap node sesuai dengan keadaan dari sistem. keadaan diwakili oleh serangkaian bit di mana setiap bit sesuai dengan keadaan flip-flop sistem.

2.10 Kebudayaan

2.10.1 Pengertian Kebudayaan

Menurut Rafiek (2012,pp7-8), Kebudayaan adalah segala daya dan aktivitas manusia untuk mengolah dan mengubah alam. Keseluruhan ide-ide,tindakan atau hasil karya manusia dalam rangka kehidupan masyarakat yang dijadikan milik diri manusia dengan belajar atau keseluruhan dari kelakukan dan hasil kelakuan itu didapat dengan cara belajar.

Kebudayaan didefiniskan sebagai cara hidup manusia yang dirancang sebagai pedoman hidupnya. Kebudayaan adlaah keseluruhan system gagasa, tindakan dan hasil karya manusia untuk memenuhi kehiduapan dengan cara belajar, yang kesemuanya tersusun dalam kehiduapan masyarakat. Kebudayaan adalah segala ciptaan manusia yang sesungguhnya merupakan usaha dan member bentuk serta susunan baru alam pemberian Tuhan sesuai dengan kebutuhan jasmani dan rohani.

2.10.2 Kriteria Kebudayaan :

1. Sesuatu yang harus ditemukan sebagai sesuatu yang baru yang sebelumnya tidak ada,
2. Sesuatu yang harus dialihkan dari generasi ke generasi,
3. Sesuatu yang harus diabadikan dalam keasliannya atau dalam bentuk yang dimodifikasi. Rafiek (2012,p11)

2.10.3 Unsur – Unsur Kebudayaan

1. kegiatan kebudayaan,
2. Kegiatan kebudayaan sebagai kompleks kebudayaan,
3. Integrasi kebudayaan.

Kebudayaan dapat dibagi menjadi 3 bagian utama, yaitu:

1. Adat Istiadat

   Adat istiadat, yaitu “kelompok kebiasaan”. Kebiasaan-kebiasaan itu adalah cara yang sesungguhnya anggota kelompok berinteraksi atau bertingkah laku. Contohnya adalah cara berkomunikasi, cara supaya tetap bersih, dan cara makan makanan yang khas.

2. Sistem Gagasan

   Sistem gagasan adalah seperangkat ide yang menetapkan standar tingkah laku yang baik dan buruk, serta memberikan makna dan maksud hidup. Contohnya religi dan norma yang menetapkan cara seseorang harus bertingkah laku. Norma adalah standar cara harus bertingkat laku, yang pada hakikatnya mungkin bersumber pada agama (religi) dan norma.

3. Benda Hasil Karya

   Benda Hasil Karya adalah obyek yang dihasilkan dan dipakai masyarakat, termasuk alat-alat yang dipakai utnuk memproduksi benda lain. Rafiek (2012,pp15-16)

2.10.4 Kebudayaan Nasional dan Daerah

Menurut Rafiek (2012,pp19-23) Kebudayaan lama dan asli yang terdapat sebagai puncak-puncak kebudayaan di daerah di seluruh Indonesia terhitung sebagai kebudayaan bangsa. Usahaa kebudayaan harus menuju ke arah kemajuan adab, budaya dan persatuan, dengan tidak memperkembangkan atau memperkaya kebudayaan bangsa sendiri, serta mempertinggi derajat kemanusian bangsa Indonesia.

Kebudayaan bangsa Indonesia mengandung prinsip-prinsip sebagai berikut :

1. Asa kekeluargaan dan musyawarah,
2. Saling memberi dan mengalah,
3. Saling Asah, Asih, dan Asuh.

Dua paham besar budaya:

1. Paham monodualistik

   Paham monodualistik adalah suatu paham yang menganggap bahwa hakikat sesuatu adalah dua unsur yang terikat menjadi suatu kebulatan.

2. Paham monopluralistik

   Paham monopluralistik adalah suatu paham yang mengakui bangsa Indonesia terdiri atas berbagai unsure yang beraneka ragam, suku bangsa, adat istiadat, budaya, kesenian, bahasa daerah, berbeda

agama, dan sebagainya, tetapi semuanya terikat menjadi suatu kesatuan.

2.10.4.1 Kebudayaan Nasional

Kebudayaan nasional merupakan kepribadian bangsa yang berasal dari pola pikir kehidupan social, atau puncak dari budaya suku-suku bangsa yang menghuni bumi nusantara ini dan hasil sintesis dari berbagai jenis budaya suku tersebut yang membentuk pola baru. Dengan kata lain, perwujudan atau perpaduan unsure-unsur kebudayaan daerah atau lapisan kebudayaan bangsa Indonesia yang mencerminkan semua aspek perikehidupan yang totalitas meliputi bahasa, kesenian, adat istiadat, tradisi, hukum, dan kepercayaan, tumbuh dan berkembang dalam masyarakat, atau apa saja yang dihasilkan, yang memperkuat kepribadian, merupakan sumber inspirasi dan kreativitas serta identitas puncak-puncak kebudayaan daerah, sedangkan budaya nasional adalah kebudayaan yang dapat mencerminkan kepribadian bangsa Indonesia dan dapat menunjukkan identitas nasional, meliputi bidang politik, ekonomi, sosial, dan budaya.

Pelestarian kebudayaan nasional dengan cara menggali, merawat, dan mengembangkan peninggalan kebudayaan yang sudah ada.

Pelestarian budaya nasional meliputi:

1. Bidang kepurbakalaan, kesejahteraan dan permuseuman,

2. Bidang seni budaya,
3. Bidang kebahasaan dan kesastraan.

Kebudayaan nasional merupakan kepribadian bangsa yang berasal dari pola pikir kehidupan sosial, contohnya:

1. Kain Batik
2. Gotong Royong
3. Ramah Tamah
4. Tarian Nasional
5. Musyawarah
6. Lagu Kebangsaan ‘Indonesia Raya’
7. Candi Borobudur
8. TMII

2.10.4.2 Kebudayaan Daerah

Kebudayaan daerah adalah kebudayaan yang tumbuh dan berkembang dalam masyarakat suku yang membedakannya dari kebudayaan suku yang lain karena factor adat, kepercayaan, agama dan lingkungan alam yang dapat bertahan karena ikatan tradisi antardaerah, yaitu melalui pertukaran misi kesenian, melalui komunikasi media massa. Ciri kebudayaan daerah adalah unsur tradisi yang berakar kuat pada masyarakat kesukuan. Kebudayaan daerah dapat kebudayaan daerah dapat diangkat menjadi kebudayaan nasional harus mempunyai sifat:

1. Harus bersifat khas, maksudnya harus pantas dan tepat sebagai kebudayaan nasional.
2. Harus mampu member cirri sebagai kepribadian atau identitas bangsa.
3. Harus bermutu tinggi sehingga dapat menjadi kebanggaan seluruh rakyat.

   Contohnya:

   1. Rumah atau corak bangunan adat,
   2. Pakaian Adat,
   3. Tarian daerah,
   4. Lagu daerah,
   5. Adat perkawinan,
   6. Makanan khas daerah,
   7. Alat music tradisional,
   8. Seni sastra daerah,
   9. Bahasa daerah,
   10. Hukum adat,
   11. Kerajinan tangan daerah,
   12. Tenunan khas daerah,
   13. Kekayaan ilmu gaib(mistik).