BAB II
TINJAUAN PUSTAKA

Pada bagian tinjauan pustaka akan diuraikan berbagai teori atau referensi yang terkait dan menunjang permasalahan yang akan diteliti. Referensi tersebut berkaitan dengan sistem pakar, penyebab serta solusi kerusakan mesin, dan referensi lainnya. Bab ini bertujuan untuk mendukung permasalahan yang akan diteliti serta mendukung hasil penelitian.

2.1 PENELITIAN TERDAHULU

Ada beberapa penelitian sebelumnya yang berkaitan dengan perancangan *computerized maintenance management system* mesin *bending*. Berikut penjelasan singkat beberapa penelitian tersebut:

1. Yudistira (2012). Penelitian yang dilakukan di PT. Sekar Laut Tbk. di Sidoarjo ini membahas masalah yang terjadi akibat besarnya kapasitas produksi sehingga kebutuhan akan adanya pemeliharaan mesin-mesin produksi juga semakin tinggi. Perusahaan juga tidak mendokumentasikan data-data mengenai mesin, karyawan, dan *spare parts* bahkan belum ada tindakan pemeliharaan pencegahan karena selama ini pemeliharaan hanya dilakukan ketika mesin mengalami kerusakan. Metode yang digunakan untuk mengembangkan sistem informasi pemeliharaan *maintenance* meggunakan metode SDLC (*System Development Life Cycle*) dan *software* yang digunakan untuk membuat *prototype* adalah VB.Net. Hasil yang diperoleh menunjukkan bahwa sistem informasi yang dirancang telah mampu melakukan proses penyimpanan beberapa data dan dapat menentukan penjadwalan *maintenance* mesin secara keseluruhan 30 hari ke depan. Proses yang terkomputerisasi ini sangat berguna untuk mengurangi hilangnya data tentang pemeliharaan mesin sehingga dapat membantu perusahaan dalam melakukan pemeliharaan mesin produksi.
2. Nilmada (2013) merancang dan membuat aplikasi sistem pakar untuk mendeteksi kerusakan pada sepeda motor secara cepat dan akurat dengan menggunakan bahasa pemrograman *visual basic*. Metode penelusuran yang digunakan adalah *forward chaining* karena pelacakan dimulai dari keadaan awal dengan melihat hasil analisis gejala-gejala kerusakan yang ada lalu dicocokkan dengan tujuan yang diharapkan.

Aplikasi ini bukan bertujuan menggantikan kerja para mekanik namun berfungsi sebagai asisten yang handal dalam membantu kerja mekanik karena konsistensi dalam pengetahuan dan diagnosa yang diberikan. Kehandalan aplikasi ini juga bergantung dari penambahan pengetahuan yang dilakukan baik oleh *programmer* maupun pakar terkait.

3. Arifin (2011) merancang dan membangun sebuah *prototype* sistem pakar untuk pengendalian kualitas produk proten. Sistem berbasis *website* ini terdiri dari tiga komponen utama yaitu basis pengetahuan, mesin inferensi, dan penghubung antarmuka (*user interface*). Metode sistem pakar yang digunakan adalah *backward chaining* guna mengetahui penyebab kerusakan produk proten yang diproduksi oleh perusahaan dimana *prototype* aplikasi terdiri dari tiga bagian, yaitu *welcome screen*, jendela konsultasi, dan jendela rekomendasi.
4. Reyhan (2013) melakukan analisis faktor penyebab kecelakaan kerja proyek konstruksi GKM tower menggunakan metode *fault tree analysis* (FTA) untuk menyelesaikan kasus apabila terjadi suatu kegagalan atau hal yang tidak diinginkan dengan mencari akar-akar permasalahan *basic events* yang muncul dan diuraikan dari setiap indikasi kejadian puncak (*top event*). FTA mengidentifikasi hubungan antara faktor penyebab dan ditampilkan dalam bentuk pohon kesalahan yang melibatkan gerbang logika sederhana. Namun dalam aplikasinya dalam permasalahan konstruksi, FTA tidak mudah dilakukan. Hal ini disebabkan dibutuhkannya banyak data dan dokumentasi yang berhubungan dengan kecelakaan kerja dengan baik. Data dan dokumentasi terkait kecelakaan kerja yang baik dibutuhkan untuk memperoleh gambaran atau model konstruksi FTA yang baik, sehingga faktor penyebab kecelakaan kerja yang terjadi di proyek konstruksi dapat dianalisis dengan baik. Untuk mendapatkan data yang baik, dibutuhkan keterbukaan dari pihak kontraktor. Selain itu, dibutuhkan dokumentasi kecelakaan kerja yang baik yang seharusnya dilakukan oleh pihak kontraktor, untuk setiap jenis kecelakaan kerja, baik dengan risiko kecil maupun risiko besar. Untuk mencegah dan menangani faktor penyebab kecelakaan kerja tersebut, tentu diperlukan suatu manajemen K3 yang berlaku pada proyek tersebut.

Penelitian ini dilakukan di departemen *maintenance* PT. Dinamika Energitama Nusantara dimana ada permasalahan mengenai belum adanya sistem untuk *troubleshooting* mesin yang menyebabkan karyawan *maintenance* mengalami kesulitan ketika menangani kerusakan mesin karena selama ini hanya berdasarkan pengalaman

teknisi dan pencarian solusi dengan *manual book* mesin. Selain itu juga belum ada sistem informasi yang menyediakan jadwal untuk melakukan *preventive maintenance* sehingga seringkali terjadi keterlambatan pemeliharaan mesin yang berakibat terjadinya kerusakan mesin yang seharusnya bisa dicegah. Tabel 2.1 merupakan tabel perbandingan penelitian pendahuluan dengan penelitian yang akan dilakukan.

Tabel 2.1 Komparasi Penelitian dengan Penelitian Terdahulu

Peneliti	Objek	Metode	Hasil Penelitian
Yudistira (2012)	Mesin-mesin produksi	SDLC (*System Development Life Cycle*)	Sistem informasi yang memuat data mesin, karyawan, *sparepart*, jenis *maintenance*, kerusakan mesin, dan menentukan jadwal *maintenance* 30 hari ke depan.
Nilmada (2013)	Sepeda motor	*Inference engine forward chaining*	Sistem pakar untuk diagnosa kerusakan sepeda motor dengan memanfaatkan *database microsoft access* dengan bahasa pemrograman *visual basic* untuk menangani *troubleshooting* berdasarkan kemungkinan penyebab kerusakan.
Arifin (2011)	Produk proten	*Inference engine backward chaining*	Sistem pakar berbasis web guna mengetahui penyebab kerusakan produk proten dengan metode yang digunakan adalah *backward chaining*.
Reyhan (2013)	Proyek konstruksi GKM *tower*	*Fault tree analysis*	Mengetahui dan menganalisis faktor penyebab kecelakaan kerja proyek konstruksi dengan metode *fault tree analysis*.
Firda (2015)	Mesin *bending* departemen *maintenance* PT. Dinamika Energitama Nusantara	*Software prototyping* dengan *Fault Tree Analysis* dan *Decision Table*	Merancang dan membangun *computerized maintenance management system* dengan mengetahui dan menganalisis faktor penyebab kerusakan mesin *bending* menggunakan metode *fault tree analysis* yang ditransformasikan kedalam *decision table* serta menentukan jadwal *maintenance* komponen mesin *bending* berdasarkan perhitungan *mean time between failure* (MTBF).

2.2 PERAWATAN (*MAINTENANCE*)

2.2.1 Manajemen Perawatan

Manajemen perawatan didefinisikan sebagai organisasi pemeliharaan yang sesuai dengan kebijaksanaan yang disetujui. Kebijaksanaan yang disetujui harus sejelas mungkin dan tidak boleh meragukan. Hal ini merupakan tanggungjawab tim manajemen puncak untuk menentukannya. Kebijaksanaan ini juga harus mendefinisikan kondisi perawatan yang bisa diterima dan manajer perawatan harus diberi tahu mengenai kebijaksanaan ini. Uraian pekerjaan harus meliputi suatu pernyataan kebijaksanaan pemeliharaan sebagaimana telah ditetapkan oleh manajemen, dan ini harus menjadi batas persyaratan baginya (Corder, 1992).

Perawatan mesin merupakan suatu faktor yang memegang kendali penting dalam suatu industri guna menjaga kestabilan kondisi mesin/fasilitas produksi agar dapat beroperasi dengan baik, sehingga dapat meminimalkan adanya *breakdown.*

2.2.2 Pengertian dan Tujuan Perawatan

Perawatan merupakan kegiatan untuk menjaga atau memelihara fasilitas-fasilitas dan peralatan pabrik, serta mengadakan perbaikan, penyesuaian, atau penggantian yang diperlukan untuk mendapatkan suatu kondisi operasi produksi yang memuaskan dan sesuai dengan yang direncanakan. Disamping itu ada juga yang mendefinisikan perawatan sebagai suatu konsepsi dari semua aktivitas yang diperlukan untuk menjaga ataupun mempertahankan kualitas peralatan agar tetap dapat berfungsi dengan baik seperti dalam kondisi yang sebelumnya (Supandi, 1990: 26).

Menurut Corder (1992), tujuan perawatan dapat didefinisikan sebagai berikut:

1. Memperpanjang usia kegunaan aset yaitu setiap bagian dari suatu tempat kerja, bangunan, dan isinya.
2. Menjamin ketersediaan optimum peralatan yang dipasang untuk produksi atau jasa dan mendapatkan laba investasi (*return of investment*) maksimum yang mungkin.
3. Menjamin kesiapan operasional dari seluruh peralatan yang diperlukan dalam kegiatan darurat setiap waktu, seperti unit cadangan, unit pemadam kebakaran dan penyelamat dan sebagainya.
4. Menjamin keselamatan orang yang menggunakan sarana tersebut.

2.2.3 Jenis-jenis Perawatan

Bentuk-bentuk perawatan (Supandi:1990) dibagi menjadi dua kelompok yaitu :

1. Perawatan Preventif (*Preventive Maintenance*).

   Pekerjaan perawatan yang bertujuan untuk mencegah terjadinya kerusakan, atau cara perawatan yang direncanakan untuk pencegahan (preventif). Perawatan preventif dimaksudkan juga untuk mengefektifkan pekerjaan inspeksi, perbaikan kecil, pelumasan dan *set up* sehingga peralatan atau mesin-mesin selama beroperasi dapat terhindar dari kerusakan. Perawatan preventif dilaksanakan sejak awal sebelum terjadi kerusakan. Perawatan preventif ini penting diterapkan pada industri-industri yang proses produksinya kontinyu atau memakai sistem otomatis. Kegiatan *preventive maintenance* dibagi menjadi dua kelompok :

   a. *Subjective Monitoring*

      Monitoring yang dilakukan dengan menggunakan indera seperti mendengarkan, melihat, menyentuh, merasakan, dan membaui, kemudian mengestimasi kondisi berdasarkan indera tersebut. Perawatan ini bersifat subjektif karena bergantung pada keahlian operator dalam memonitor kondisi mesin.

   b. *Objective Condition Monitoring*

      Monitoring yang dilakukan berdasarkan hasil yang ditunjukkan oleh alat ukur. Pada metode ini perawatan dilakukan dengan cara memasangkan alat ukur pada peralatan/mesin yang tidak sedang beoperasi, kemudian sensor dari alat ukur tersebut akan memberikan informasi bila terjadi penyimpangan.

2. Perawatan Korektif (*Corrective Maintenance*).

   Pekerjaan perawatan yang dilakukan untuk memperbaiki dan meningkatkan kondisi fasilitas sehingga mencapai standar yang dapat diterima. Perawatan korektif termasuk dalam cara perawatan yang direncanakan untuk perbaikan. Dalam perawatan ini dapat mengadakan peningkatan-peningkatan sedemikian rupa, seperti melakukan perubahan atau modifikasi rancangan peralatan agar lebih baik. Menghilangkan problema yang merugikan untuk mencapai kondisi operasi yang lebih ekonomis.

   a. Perawatan Berjalan (*Running Maintenance*).

      Perawatan yang dilakukan pada saat fasilitas atau peralatan dalam keadaan bekerja. Perawatan berjalan ini termasuk cara perawatan yang direncanakan untuk diterapkan pada peralatan dalam keadaan operasi. Perawatan dalam

kondisi berjalan diterapkan pada mesin-mesin yang harus beroperasi terus menerus dalam proses produksi. Kegiatan perawatan monitoring secara aktif. Diharapkan dari hasil dari perbaikan yang dilakukan secara cepat dan terencana ini dapat menjamin kondisi proses produksi tanpa adanya gangguan yang mengakibatkan kerusakan.

b. Perawatan Prediktif (*Predictive Maintenance*)

Perawatan prediktif dilakukan untuk mengetahui terjadinya perubahan atau kelainan dalam kondisi fisik maupun fungsi dari sistem peralatan. Biasanya perawatan prediktif dilakukan dengan bantuan panca indera atau dengan alat-alat monitor canggih. Teknik-teknik dan alat bantu yang dipakai dalam memonitor kondisi ini adalah untuk efisiensi kerja agar kelainan yang terjadi dapat diketahui dengan cepat dan tepat. Perawatan dengan sistem monitoring sangat penting dilakukan untuk mendapatkan hasil yang realistis tanpa melakukan pembongkaran total untuk menganalisisnya.

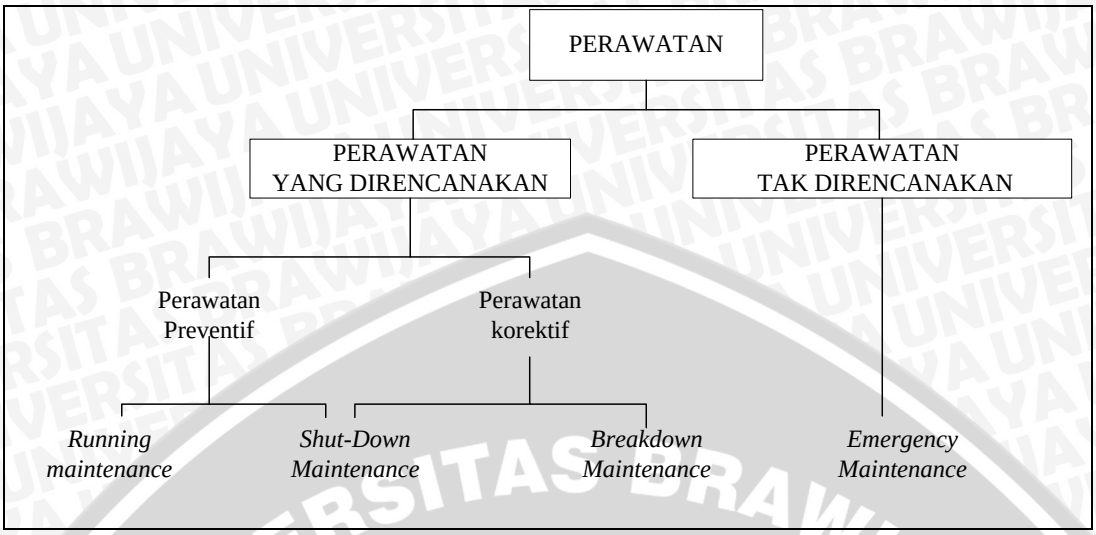
c. Perawatan Setelah Terjadi Kerusakan (*Breakdown Maintenance*)

Perawatan ini dilakukan setelah terjadi kerusakan, dan untuk memperbaikinya harus disiapkan suku cadang, material, alat-alat dan tenaga kerjanya. Beberapa peralatan pabrik yang beroperasi pada unit tersendiri atau terpisah dari proses yang lainnya, tidak akan langsung mempengaruhi seluruh proses produksi apabila terjadi kerusakan. Untuk peralatan tersebut tidak perlu diadakan perawatan , karena biaya perawatan lebih besar daripada biaya kerusakannya. Dalam kondisi khusus ini peralatan dibiarkan beroperasi sampai terjadi kerusakan, sehingga waktu untuk produksi tidak berkurang. Penerapan sistem perawatan ini dilakukan pada mesin-mesin industri yang ringan, apabila terjadi kerusakan dapat diperbaiki dengan cepat.

d. Perawatan Darurat (*Emergency Maintenance*)

Perbaikan yang segera dilakukan karena terjadi kemacetan atau kerusakan yang tak terduga. Perawatan darurat ini termasuk cara perawatan yang tidak direncanakan (*unplanned maintenance*).

Gambaran hubungan masing-masing perawatan terlihat pada Gambar 2.1



Gambar 2.1 Hubungan antara Berbagai Bentuk Perawatan
Sumber: Supandi (1990)

Pada dasarnya, tidak semua bentuk perawatan cocok diterapkan pada setiap mesin atau peralatan. Oleh sebab itu, perlu dilakukan beberapa pertimbangan dalam memilih metode penentuan waktu perawatan yang tepat, karena pemilihan strategi perawatan yang sesuai dapat menghasilkan kondisi mesin/peralatan yang optimum. Terdapat beberapa faktor yang dapat dijadikan acuan dalam memilih metode perawatan yang cocok, yaitu :

1. Jumlah mesin/peralatan yang digunakan.
2. Umur masing-masing mesin yang digunakan.
3. Tingkat produksi perusahaan.
4. Tingkat keahlian teknisi yang dimiliki perusahaan.

Pelaksanaan kegiatan perawatan tidak terlepas dari penjadwalan perawatan. Penjadwalan perawatan untuk tiap komponen pada setiap mesin dapat berbeda, bergantung pada lamanya selang waktu kerusakan dan kapasitas kerja yang dimiliki mesin atau komponen yang bersangkutan.

2.2.4 *Maintenance Performance*

Maintenance performance terdiri dari 3 bagian menurut Kostas (2012:95) :

1. *Reliability*, yaitu kemungkinan (probabilitas) dimana peralatan dapat beroperasi dibawah keadaan normal dengan baik. *Mean Time Between Failure (MTBF)* adalah rata–rata waktu suatu mesin dapat dioperasikan sebelum terjadinya kerusakan. MTBF ini dirumuskan sebagai hasil bagi dari total waktu pengoperasian mesin

dibagi dengan jumlah/frekuensi kegagalan pengoperasian mesin karena *breakdown*. Rumus perhitungan MTBF dapat dilihat pada persamaan 2-1 dibawah ini :

$$MTBF = \frac{Total\ operation\ time}{Frekuensi\ breakdown} \quad (2\text{-}1)$$

Sumber: Kostas (2012:95)

2. *Maintainability*, yaitu suatu usaha dan biaya untuk melakukan perawatan (pemeliharaan). Suatu pengukuran dari *maintainability* adalah *Mean Time To Repair (MTTR)*, tingginya MTTR mengindikasikan rendahnya *maintainability*. Dimana MTTR merupakan indikator kemampuan (skill) dari operator *maintenance* mesin dalam menangani atau mengatasi setiap masalah kerusakan. Rumus perhitungan MTTR dapat dilihat pada persamaan 2.2 :

$$MTTR = \frac{Breakdown\ time}{Frekuensi\ breakdown} \quad (2\text{-}2)$$

Sumber: Kostas (2012:95)

Dimana *Breakdown Time* adalah termasuk waktu menunggu untuk *repair*, waktu yang terbuang untuk melakukan *repair*, waktu yang terbuang untuk melakukan pengetesan, dan waktu mendapatkan peralatan yang siap untuk mulai beroperasi.

3. *Availability***,** yaitu proporsi dari waktu peralatan/mesin yang sebenarnya tersedia untuk melakukan suatu pekerjaan dengan waktu yang ditargetkan seharusnya tersedia untuk melakukan suatu pekerjaan. Atau dengan definisi lain bahwa *availability* adalah *ratio* untuk melihat *line stop* ditinjau dari aspek *breakdow*n saja. Persamaan perhitungan dari *availability* (A) ditunjukkan pada persamaan 2-3 :

$$A = \frac{Total\ operation\ time}{Loading\ time}\ x\ 100\% \quad (2\text{-}3)$$

Sumber: Kostas (2012:95)

2.2.5 *Computerized Maintenance Management System* (CMMS)

Computerized Maintenance Management System (CMMS) adalah sebuah program komputer yang dirancang untuk membantu dalam perencanaan, manajemen, dan fungsi administratif yang dibutuhkan dalam pemeleiharaan yang efektif. Hal-hal yang termasuk ke dalam fungsi tersebut adalah membangun, merencanakan, dan melaporkan *work orders*; perkembangan dari catatan-catatan mengenai pemeliharaan yang mudah untuk dicari, dan dapat mencatat transaksi pembelian komponen (Bagadia, 2006:5).

CMMS bukan sekedar digunakan sebagai alat pengontrol sistem pemeliharaan, namun sekarang ini CMMS dapat digunakan meningkatkan kondisi peralatan dan juga *output*nya. CMMS menawarkan fungsi-fungsi dari pemeliharaan yang tidak hanya

terbatas pada hal manufaktur saja. CMMS juga dapat diaplikasikan untuk fasilitas, utilitas, dan berbagai tipe organisasi lainnya di mana peralatan digunakan sebagai subjek, dan perbaikan yang harus dilakukan terhadap peralatan- peralatan yang mengalami kerusakan. Sebuah CMMS biasanya terdiri dari *equipment management, preventive maintenance, labor, work orders, planning/ scheduling, inventory control, and purchasing.*

CMMS dapat digunakan untuk menangani berbagai macam proses dari sistem pemeliharaan, membantu perusahaan dalam membuat sistem pemeliharaan menjadi lebih efisien, dan menganalisa peralatan yang lebih jauh digunakan untuk optimasi performansi peralatan tersebut (*Mather*, 2003:2). Sebuah CMMS dasar terdiri dari: e*quipment data management, preventive maintenance, labor, work order system, scheduling /planning, vendor, inventory control, purchasing*, dan *budgeting.*

Modul-modul ini berdiri sendiri ataupun bergabung antara modul yang satu dengan yang lain. Sebagai contoh, CMMS yang menggabungkan *equipment data* dan *work orders* modul dapat dengan otomatis memasukkan informasi dari peralatan ke dalam *work orders* yang dapat dilakukan hanya dengan menginput identitas dari peralatan tersebut sehingga hasilnya akan lebih cepat dan lebih akurat. Penerapan CMMS yang baik akan dapat menghasilkan efisiensi dalam berbagai hal termasuk efisiensi dalam hal manajemen yang sangat tidak mungkin dapat dicapai tanpa menggunakan CMMS (*Smith*, 2004:78).

2.3 FAULT TREE ANALYSIS (FTA)

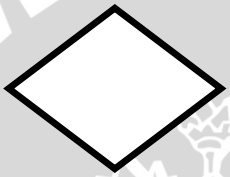
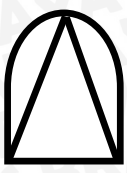
FTA adalah metode analisa, dimana terdapat suatu kejadian yang tidak diinginkan yang disebut dengan *undesired event* terjadi pada suatu sistem, dan sistem tersebut kemudian dianalisa dengan kondisi lingkungan dan operasional yang ada untuk menemukan semua cara yang mungkin menyebabkan terjadinya *undesired event* tersebut. Penguraian setiap elemen yang menyebabkan terjadi *undesired event* tersebut dilakukan dengan menggunakan pohon kesalahan. Pohon kesalahan adalah suatu model grafis yang merupakan gambaran hubungan timbal balik yang logis dari peristiwa-peristiwa dasar yang telah didefinisikan sebelumnya. Pembangunan model pohon kesalahan (*Fault Tree)* dilakukan dengan cara wawancara dengan manajemen dan melakukan pengamatan terhadap proses diinginkan. Langkah-langkah membangun FTA:

1. Mendefinisikan permasalahan.

2. Mempelajari sistem dengan cara mengetahui proses yang berlaku, lingkungan kerja dan prosedur operasi.
3. Mengembangkan pohon kesalahan.

Simbol-simbol FTA dapat dilihat pada Tabel 2.2:

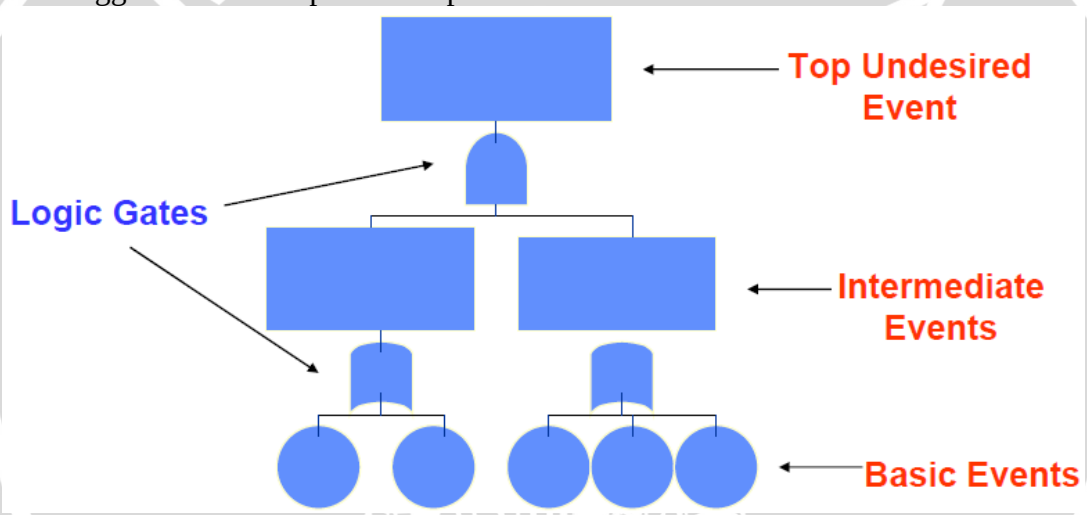
Tabel 2.2 Simbol-simbol FTA

Event		
Nama	Simbol	Keterangan
Circle		Gambar *circle* menunjukkan kejadian pada level paling bawah (*lowest level failure event*) atau disebut kejadian paling dasar (*basic event*)
elipse		Gambar *elipse* menunjukkan kejadian pada level paling atas (*top level event*) dalam pohon kesalahan
Diamond		Gambar *diamond* menunjukkan kejadian yang tidak terduga (*undeveloped event*). Kejadian ini tidak ini terlihat pada pohon kesalahan dan dianggap sebagai kejadian paling awal yang menyebabkan kerusakan.
House		Gambar *house* menunjukkan kejadian *input* (*input event*) dan merupakan kegiatan terkendali (*signal),* kegiatan ini dapat menyebabkan kerusakan.
Rectangle		Gambar *rectangle* menunjukkan kejadian pada level menengah (*intermediate fault event*) dalam pohon kesalahan
Gate		
And		*Output* kesalahan muncul ketika seluruh input kesalahan tersebut muncul
Or		*Output* kesalahan muncul ketika sedikitnya salah satu input kesalahan muncul
Exclusive or		Output kesalahan muncul jika tepat satu dari input kesalahan muncul

Transfer		
Transfer in		Mengidentifikasikan bahwa pohon kesalahan dikembangkan lebih lanjut pada kemunculan simbol penghubung *transfer out*
Transfer		
Nama	Simbol	Keterangan
Transfer out		Mengidentifikasikan bagian dari pohon harus ditambahkan pada siimbol *transfer out*

Sumber: NASA (2002)

Penggunaan FTA dapat dilihat pada Gambar 2.2 berikut ini:



Gambar 2.2 *Fault Tree Analysis*
Sumber: NASA (2002)

2.4 SISTEM *DATABASE*

Database adalah kumpulan data yang terhubung dan disimpan secara bersama pada suatu media dengan cara-cara tertentu sehingga mudah untuk digunakan dan ditampilkan kembali, dapat digunakan oleh satu atau lebih program aplikasi secara optimal, data disimpan sedemikian sehingga penambahan, pengambilan dan modifikasi data dapat dilakukan dengan mudah dan terkontrol.

2.4.1 Komponen Sistem *Database*

Di dalam suatu sistem *database* terdapat empat komponen penting yaitu:

1. Data

   Ditinjau dari jumlah pemakai, data dapat dikelompokan menjadi 2 yaitu:

   a. *Data single user*, yaitu suatu sistem *database* yang hanya dioperasikan oleh satu pemakai.
   b. *Data multi user*, yaitu satu atau lebih pemakai bisa beroperasi secara bersamaan ke dalam *database*, sehingga data yang ada dalam *database* dapat digunakan bersama dan saling terintegrasi.

2. *Hardware*

   Merupakan piranti keras yang dibutuhkan oleh manajemen *database*, yang biasanya merupakan mesin atau alat yang standar.

3. *Software*

   Merupakan piranti lunak yang menjadi penghubung antara *database* dengan pemakai yang disebut sebagai *Database Management System* (DBMS) atau *database* manajer. Tugas dari DBMS ini adalah melakukan semua kebutuhkan akses data.

4. *User*

   User atau pemakai *database* dapat dikelompokan menjadi tiga yaitu:

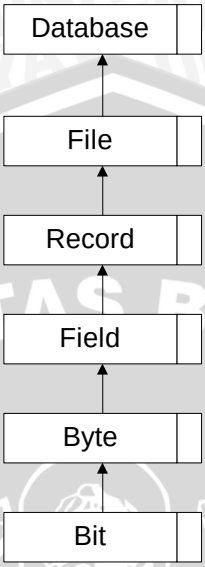
   a. *Programmer* aplikasi, yaitu orang yang bertanggung jawab untuk menulis program aplikasi.
   b. *End user*, yaitu orang yang menggunakan data dalam *database* untuk kebutuhan tugas atau fungsinya.
   c. Administrator *database*, yaitu orang yang bertanggung jawab pada keseluruhan sistem *database*.

2.4.2 Pengertian Data

Menurut kamus besar bahasa Indonesia, data diterjemahkan sebagai istilah yang berasal dari kata "*datum*" yang berarti fakta atau bahan-bahan keterangan kumpulan kejadian yang diangkat dari suatu kenyataan. Data dapat berupa angka-angka, huruf, atau simbol-simbol khusus atau gabungan darinya. Data mentah masih belum bisa bercerita banyak, sehingga perlu diolah lebih lanjut. Pengolahan data adalah manipulasi dari data ke dalam bentuk yang lebih berguna dan lebih berarti berupa suatu informasi (McLeod, 2004:167).

2.4.3 Hierarki Data

Berdasarkan tingkat kompleksitas nilai data, tingkatan data dapat disusun kedalam sebuah hirarki, mulai dari yang paling sederhana hingga paling kompleks yang dapat digambarkan dengan Gambar 2.3.



Gambar 2.3 Hierarki data
Sumber: Kusrini (2007)

Berdasarkan Gambar 2.3, pengorganisasian data dapat dibagi menjadi enam tingkatan, yaitu:

1. *Bit* adalah suatu sistem angka biner yang terdiri atas dua macam nilai saja, yaitu 0 dan 1. Sistem angka biner merupakan dasar-dasar yang dapat digunakan untuk komunikas antara manusia dan mesin (komputer) yang merupakan sekumpulan komponen elektronik dan hanya dapat membedakan dua keadaan saja (*on* dan *off*). Jadi bit adalah unit terkecil pembentuk data.
2. *Byte* adalah bagian terkecil yang dapat dialamatkan dalam memori. *Byte* merupakan sekumpulan *bit* yang secara konvensional terdiri atas delapan *bit*. Satu *byte* digunakan untuk mengkodekan satu buah karakter dalam memori. Contoh: Kode ASCII untuk J ialah 10101010. Jadi byte adalah kumpulan *bit* yang membentuk suatu karakter (huruf, angka atau tanda). Dengan kombinasi 8 *bit*, dapat diperoleh 256 karakter (= 2 dipangkat 8).
3. *Field* atau kolom adalah unit terkecil yang disebut data. *Field* merupakan sekumpulan *byte* yang mempunyai makna. Contoh: Joni yang merupakan *field* nama. Jadi *field* ibarat kumpulan karakter yang membentuk suatu kata.

4. *Record* atau baris adalah kumpulan item yang secara *logic* saling berhubungan. Setiap *record* dapat dikenali oleh sesuatau yang mengenalinya, yaitu *field* kunci. Jadi *record* ibarat kumpulan data yang membentuk satu kalimat yang berarti.
5. *File* atau tabel adalah kumpulan *record* yang sejenis dan secara logis berhubungan. Pembuatan dan pemeliharaan *file* adalah faktor yang sangat penting dalam sistem informasi manajemen yang memakai komputer. Jadi tabel ibarat kumpulan baris/*record* yang membentuk satu tabel.
6. *Database* merupakan kumpulan *file-file* yang berhubungan secara logis dan digunakan secara rutin pada operasi-operasi sistem informasi manajemen. Semua *database* umumnya berisi elemen-elemen data yang disusun ke dalam *file-file* yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di *hardware* komputer dengan *software* untuk melakukan manipulasi data untuk kegunaan tertentu. Jadi, suatu *database* adalah menunjukkan suatu kumpulan tabel yang dipakai dalam suatu lingkup perusahaan atau instansi untuk tujuan tertentu.

2.4.4 Tujuan Sistem *Database*

Tujuan utama dari sistem *database* adalah praktis dan efisien. Secara umum, tujuan dari sistem *database* adalah sebagai berikut:

1. Isolasi data, menempatkan tiap data pada tempatnya masing-masing.
2. Multi *user*, saat perusahaan mengambil pendekatan berorientasi masalah, pertama didefinisikan kemudian pengambilan keputusan untuk penyelesaian masalah tersebut dan untuk pengambilan keputusan diperlukan informasi.

Sedangkan tujuan dari manajemen sistem *database* adalah sebagai berikut:

1. Menyediakan tempat penyimpanan massal untuk data yang relevan.
2. Memudahkan pemakai dalam mengakses data.
3. Memungkinkan respon yang segera atas permintaan data dari pemakai.
4. Melakukan modifikasi terakhir dengan segera pada *database*.
5. Memungkinkan perkembangan lebih lanjut dalam sistem *database*.
6. Meminimasi duplikasi dan redudansi dalam penyimpanan data.
7. Memungkinkan secara serentak dan bersamaan beberapa pemakaian *database* yang berarti juga meningkatkan kebebasan data sehingga berguna untuk beberapa program.
8. Melindungi data dari gangguan kerusakan atau pemakaian oleh orang yang tidak terotorisasi.

2.4.5 Entity Relationship Diagram (ERD)

ERD merupakan salah satu pemodelan data konseptual yang paling sering digunakan dalam proses pengembangan *database* bertipe relasional. Model E-R adalah rincian yang merupakan representasi logika dari data pada suatu organisasi atau area bisnis tertentu. Model E-R terdiri dari beberapa komponen dasar yaitu sebagai berikut:

1. Entitas

   Entitas adalah sesuatu atau objek di dunia nyata yang dapat dibedakan dari sesuatu atau objek yang lainnya. Sebagai contoh, setiap mahasiswa dalam suatu universitas adalah suatu entitas. Setiap fakultas dalam suatu universitas adalah juga suatu entitas. Dapat dikatakan bahwa entitas bisa bersifat konseptual/abstrak atau nyata hadir di dunia nyata.

2. Hubungan antar relasi (*Relationship*)

   Hubungan antar relasi adalah hubungan antara suatu himpunan entitas dengan himpunan entitas yang lainnya. Misalnya, entitas mahasiswa memiliki hubungan tertentu dengan entitas matakuliah (mahasiswa mengambil matakuliah). Pada penggambaran model E-R, relasi adalah perekat yang menghubungkan suatu entitas dengan entitas yang lainnya. Tipe-tipe relasi tersebut adalah:

   a. Relasi satu ke satu (*one-to-one*)
   b. Relasi satu ke banyak (*one-to-many*)
   c. Relasi banyak ke banyak (*many-to-many*)

3. Kardinalitas/Derajat Relasi

   Kardinalitas relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Sebagai contoh: entitas-entitas pada himpunan entitas mahasiswa dapat berelasi dengan satu entitas, banyak entitas atau tidak satupun entitas dari himpunan entitas kuliah. Tabel 2.2 menjelaskan macam-macam relasi yang ada dalam kardinalitas, berikut penjelasan dari masing-masing relasi yang ada.

   a. Nol ke satu (*Zero to One*), nol atau setiap anggota entitas A hanya boleh berhubungan dengan satu anggota entitas B, dan sebaliknya.
   b. Nol ke banyak (*Zero to Many*), nol atau setiap anggota entitas A dapat berhubungan dengan dengan lebih dari satu anggota entitas B, dan sebaliknya.
   c. Satu ke satu (*One to One*), Setiap anggota entitas A hanya boleh berhubungan dengan satu anggota entitas B, begitu pula sebaliknya.

d. Satu ke banyak (*One to Many*), Setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya.

Berikut merupakan Tabel 2.3 yang menjelaskan kriteria masing-masing kardinalitas pada ERD yang telah dijelaskan.

Tabel 2.3 Kardinalitas pada Relasi Dua Entitas dalam ERD

No.	Notasi	Derajat Relasi Maks-Min	Nama Relasi	Keterangan
1.		(0 , 1)	*Zero to One*	Relasi satu dengan salah satu atribut boleh tidak berelasi (relasi kosong)
2.		(0 , N)	*Zero to Many*	Relasi banyak dengan salah satu atribut dapat tidak berelasi
3.		(1 , 1)	*One to One*	Relasi satu dengan salah satu atribut harus berelasi
4.		(1 , N)	*One to Many*	Relasi banyak dengan salah satu atribut harus mempunyai relasi

Sumber: Lio (2007)

2.4.6 Normalisasi

Perancangan *database* yang efisien sangat berguna dalam menghemat penggunaan ruang penyimpanan, kecepatan pengambilan data, dan kemudahan dalam memanipulasi data. Oleh karena itu dibutuhkan proses untuk mengubah suatu bentuk struktur data menjadi lebih sederhana dan stabil disebut dengan normalisasi. Proses dalam menganalisa data dalam normalisasi dilakukan dengan beberapa tahapan sehingga terbentuk *normal form*. Untuk menciptakan relasi *normal form* dapat dibentuk dengan mengaplikasikan suatu aturan sederhana tanpa mengurangi fungsi ketergantungan (relasi antara atribut) dalam relasi. Berikut ini diberikan gambaran tentang aturan tersebut secara ringkas.

1. *First normal form* (1NF).
   Menghapus semua elemen yang berulang sehingga hanya terdapat satu nilai untuk setiap perpotongan baris dan kolom pada tabel.
2. *Second normal form* (2NF)
   Setiap atribut bukan kunci (*non key*) yang terdapat pada relasi harus tergantung pada *primary key*. Jika hal ini tidak terpenuhi maka perlu dibuat tabel baru.
3. *Third normal form* (3NF)
   Jika terdapat saling ketergantungan antar dua atribut yang bukan merupakan atribut kunci, maka harus dibuat tabel baru. Dengan kata lain setiap relasi (tabel) hanya memuat satu kepentingan.

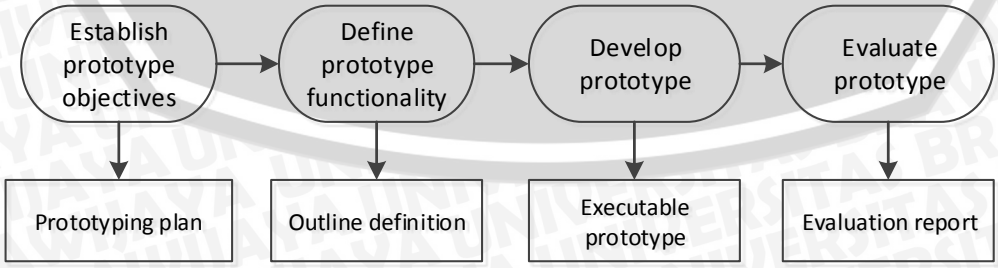
2.5 SOFTWARE PROTOTYPING

Menurut Kendall (2007), *prototyping* adalah suatu teknik pengumpulan data yang sangat berguna melengkapi siklus hidup pengembangan sistem tradisional. Saat penganalisis sistem menggunakan *prototyping*, mereka berusaha mencari reaksi, saran – saran, inovasi, rencana revisi pengguna untuk membuat peningkatan terhadap *protoype* sekaligus memodifikasi rencana sistem dengan biaya dan gangguan maksimum.

Istilah *prototyping* memiliki beberapa arti berbeda, dimana empat diantaranya paling banyak digunakan. Definisi *protoype* pertama adalah penyusunan *protoype patch*. Definisi kedua adalah *protoype* operasional yang digunakan untuk menguji fitur – fitur desain tertentu. Konsepsi ketiga adalah menciptakan *protoype* pertama yang benar – benar operasional. Sedangkan jenis *protoype* yang keempat adalah *protoype* fitur – fitur terpilih yang memiliki beberapa fitur – fitur sistem inti. Empat petunjuk utama untuk mengembangkan suatu *protoype* adalah:

1. Bekerja menurut modul – modul dapat dikendalikan
2. Membangun *protoype* dengan cepat
3. Memodifikasi *protoype*
4. Menekankan *interface* pengguna.

Sommerville (2011) menjelaskan bahwa terdapat empat tahapan dalam pegembangan *software* menggunakan *prototyping*. Langkah pertama yaitu menetapkan tujuan *protoype* dengan mengidentifikasi masalah pada sistem yang akan dibuat *prototype*-nya. Langkah kedua yaitu mendefinisikan fungsi *protoype* sehingga sesuai dengan kebutuhan *user*. Langkah ketiga yaitu mengembangkan *protoype* dengan merancang desain logis hingga implementasi konsep yang telah dibuat. Langkah terakhir yaitu evaluasi *protoype* dengan membandingkan hasil *protoype* dan kebutuhan *user*. Langkah-langkah pengembangan tersebut dapat dilihat pada Gambar 2.4.



Gambar 2.4 Model Proses Pengembangan *Protoype*
Sumber : Sommerville (2011)

Berdasarkan Gambar 2.4 model proses pengembangan *prototype* yang lebih rinci dapat dijelaskan sebagai berikut:

1. Menetapkan tujuan *prototype*
   a. Mengidentifikasi masalah dalam sistem yang sedang berjalan dengan menggunakan analisis PIECES *(Performance-Information-Economy-Control-Eficiency-Services)*
   b. Menetapkan batasan-batasan (*constraints*) atau ruang lingkup *prototype* yang akan dirancang.
   c. Menetapkan tujuan dan manfaat dari *prototype*
2. Mendefinisikan fungsi *prototype*
   a. Membuat model kebutuhan sistem (*Requirement Modelling*)

      Suatu kebutuhan sistem dinyatakan dalam suatu *checklist* yang disebut SRC (*System Requirement Checklist*). SRC adalah fitur-fitur atau karakteristik yang harus ada dalam sistem informasi untuk memenuhi kebutuhan bisnis (*business requirements*) dan yang dapat diterima pengguna. Model kebutuhan sistem ini digambarkan ke dalam lima kategori umum yaitu *output*, *input*, *process*, *performance*, dan *control*.
   b. Membuat model data (*Data Modelling)*

      Pada *data modelling*, sistem yang akan dirancang digambarkan dengan aliran data dan informasi yang dihasilkan dari dan untuk *entity* dalam sistem dengan *Data Flow Diagram* agar nampak jelas.
   c. Membuat model proses (*Process Modelling*)

      Menggambarkan logika atau aturan bisnis yang dapat dinyatakan dengan *flowchart* atau tabel aturan. Model proses nanti akan berguna dalam membangun algoritma program komputer.
   d. Pengembangan Strategi (*Development Strategies*)

      Development Strategies atau strategi pengembangan adalah merupakan tahap untuk menggambarkan kegiatan yang tersisa dalam tahap analisis sistem. Tahap ini menjelaskan transisi dari analisa sistem ke sistem desain, *prototyping*, dan pedoman desain sistem diakhiri dengan bagaimana pengembangan perangkat lunak yang dirancang.
3. Mengembangkan *prototype*
   a. Langkah Desain
      1) Desain *Database* Logis

Konsep model data yang telah terbentuk pada tahap *data modelling* dibawa ke bentuk yang logis. Desain logis terdiri dari daftar entitas untuk pembuatan tabel dan normalisasi tabel.

2) Desain *Database* Fisik

Desain fisik merupakan aktualisasi dari desain logis. Disini *entity* sudah berubah menjadi *table* dengan rancangan bentuk fisik.

3) Desain *User Interface* (UI), bertujuan untuk membuat rancangan dari tampilan sistem yang nantinya akan berinteraksi langsung dengan *user* (pengguna). Desain *user interface* harus siap digunakan dan sesuai dengan kebutuhan pengguna.

4) Desain Algoritma, bertujuan untuk merancang tahapan proses apa saja yang harus dilakukan sehingga *input, user interface,* dan *database* menghasilkan *output* yang diharapkan dan dapat ditampilkan, algoritma dapat dinyatakan dengan *flowchart* ataupun *pseudocode.*

b. Implementasi, langkah ini adalah membuat aplikasi pada tingkatan *prototype* dari spesifikasi dan konsep desain yang dirancang dengan melakukan pengembangan *database, module* dan *user interface*.

4. Mengevaluasi *prototype*

Pada langkah ini dilakukan pengujian terhadap program aplikasi yang telah dibuat. Pengujian program ini ditinjau dari tiga segi, yaitu verifikasi, validasi, dan uji *prototype*.

a. Verifikasi, menguji apakah *prototype* berjalan sesuai yang telah direncanakan. Uji Verifikasi ini meliputi pengujian *hierarki menu*, *form*, *report* dan pengujian ketelitian.

b. Validasi, menguji apakah fungsi *prototype* yang dirancang telah merepresentasikan kebutuhan user yang meliputi lima kategori umum: *output, input*, proses, *performance,* dan *control* (SRC).

c. Uji *prototype* bertujuan untuk mengetahui apakah *prototype* dapat mengatasi masalah dan kelemahan sistem yang lama.

Menurut Wetherbe (1994), untuk mengidentifikasi dan mengevaluasi permasalahan menggunakan *software protoyping* dapat menggunakan PIECES (*Performance, Information*, *Economy*, *Control*, *Efficiency*, dan *Services*).

1. *Performance* (kinerja) adalah suatu kemampuan sistem dalam menyelesaikan tugas dengan cepat sehingga sasaran dapat segera tercapai. Kinerja diukur dengan jumlah

produksi (*throughput*) dan waktu yang digunakan untuk menyesuaikan perpindahan pekerjaan *(response time*).

2. *Information* (informasi) merupakan hal penting karena dengan informasi tersebut pihak manajemen dapat melakukan pengambilan keputusan dan pengguna dapat melakukan langkah selanjutnya. Apabila kemampuan sistem baik, maka pengguna akan mendapatkan informasi yang akurat, tepat waktu dan relevan sesuai dengan yang diharapkan.
3. *Economy* (ekonomi) merupakan penilaian sistem atas penghematan dan keuntungan yang akan didapatkan dari sistem yang dikembangkan. Peningkatan terhadap kebutuhan ekonomis mempengaruhi pengendalian biaya dan peningkatan manfaat.
4. *Control* (kendali) dipasang untuk meningkatkan kinerja sistem, mencegah atau mendeteksi kesalahan sistem, menjamin keamanan data, informasi dan kebutuhan.
5. *Efficiency* (efisiensi) menyangkut bagaimana menghasilkan *output* sebanyak-banyaknya dengan *input* yang sekecil mungkin. Sistem dapat dikatakan tidak efisien bila banyak waktu yang terbuang pada aktivitas sumber daya manusia, mesin dan komputer, peng-*input*-an data yang berlebihan, pemrosesan data yang berlebihan, atau informasi yang dihasilkan berlebihan.
6. *Service* (pelayanan) menyangkut penilaian dari suatu sistem yang dilihat pula dari kriteria-kriteria seperti keakuratan dan konsistensi produk yang dihasilkan sistem, kemudahan sistem untuk dipelajari dan digunakan, atau fleksibilitas.

2.6 VISUAL BASIC FOR APPLICATION (VBA)

VBA menurut MacDonald (2010) adalah bahasa pemrograman untuk *Microsoft Office* dan aplikasi yang terkait. Dalam VBA juga dapat menggunakan *macro* untuk membuat objek dalam aplikasi sistem yang koheren. Dengan menggunakan *macro* dapat membantu menyelesaikan tugas – tugas kecil dalam aplikasi *Microsoft.* VBA digunakan karena beberapa alasan, diantaranya adalah:

1. Membuat aplikasi lebih mudah dipertahankan.
2. Dapat membuat fungsi sesuai keinginan.
3. Membuat atau memanipulasi objek.
4. Melakukan sistem berbasis pada tindakan.
5. Memanipulasi *record* pada suatu waktu.

2.7 MICROSOFT ACCESS

Microsoft Access merupakan program manajemen *database* yang relatif mudah untuk dipelajari karena dalam *Microsoft Access* diberi banyak sekali fasilitas yang membantu pemakai. Selain itu *Wizard* atau tuntunan tahap demi tahap disediakan oleh *Access* untuk merancang *tabel, query, form* dan *report* dengan cepat dan mudah. Dan tentu saja keuntungan utama dari penggunaan *Microsoft Access* adalah kemampuaan integrasinya yang optimal dengan aplikasi *software* lainnya (terutama *office*). Keunggulan ini disebabkan oleh suatu sarana penunujang yaitu *Visual Basic for Application.* Berikut ini diberikan beberapa istilah yang digunakan dalam *Microsoft Access*, yaitu:

1. *Table,* merupakan komponen utama dari sebuah *database* sekaligus obyek pertama yang harus dibuat. Tabel yang dibuat dapat berjumlah satu atau lebih sesuai dengan kebutuhan.
2. *Queries,* bagian ini digunakan untuk mengatur data mana saja dari suatu tabel yang perlu ditampilkan. Dalam *query* ini juga dapat diatur kriteria atau syarat penampilan suatu data serta bagaimana data tersebut diurutkan.
3. *Form,* bagian ini digunakan untuk memasukan data ke dalam tabel.
4. *Report,* bagian ini digunakan untuk membuat laporan dari suatu data yang telah diolah menjadi informasi. Dari *report* dapat langsung dibuat *print out* informasi yang telah diolah.

2.8 *DECISION TABLE*

Decision table merupakan suatu metode yang digunakan untuk menjelaskan dan menggambarkan aliran data secara logika yang tersimpan didalamnya yang dapat digunakan untuk menyelesaikan sebuah masalah (Vanthienen, 1994). *Decision table* bekerja dengan cara mengkombinasikan semua kondisi yang ada dimana kondisi ini berisikan aturan-aturan (rules) yang disimpan dalam bentuk tabel pada suatu masalah sehingga dapat dipastikan bahwa tidak ada kemungkinan yang terlewati didalam analisa logika terhadap masalah tersebut.

Langkah pembuatan *decision table* adalah sebagai berikut:

1. Menentukan kondisi yang akan diseleksi
2. Menentukan jumlah kemungkinan kejadian muncul
3. Menentukan tindakan yang akan dilakukan