BAB II
LANDASAN TEORI

2.1 Rancang Bangun

Menurut (Pressman 2010) “Rancang merupakan serangkaian prosedur untuk menerjemahkan hasil analisa dari sebuah sistem kedalam bahasa pemrograman untuk mendeskripsikan dengan detail bagaimana komponen-komponen sistem diimplementasikan”. Rancangan sistem adalah penentuan proses dan data yang diperlukan oleh sistem baru. Perancangan adalah kegiatan yang memiliki tujuan untuk mendesain sistem baru yang dapat menyelesaikan masalah-masalah yang dihadapi perusahaan yang diperoleh dari pemilihan alternatif sistem yang terbaik.

Bangun atau pembangunan sistem adalah kegiatan menciptakan sistem baru maupun mengganti atau memperbaiki sistem yang telah ada baik secara keseluruhan maupun sebagian. Bangun sistem adalah membangun sistem informasi dan komponen yang didasarkan pada spesifikasi desain. Dengan demikian pengertian rancang bangun merupakan kegiatan menerjemahkan hasil analisa ke dalam bentuk paket perangkat lunak kemudian menciptakan sistem tersebut ataupun memperbaiki sistem yang sudah ada.

2.2 Pengertian Sistem

Menurut (Jogiyanto 2005) “Sistem adalah suatu jaringan kinerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu. Pendekatan sistem yang lebih menekankan pada elemen atau komponen. Mendefinisikan sistem sebagai kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu”. Dengan demikian maka sistem dapat dikatakan suatu prosedur yang saling berhubungan dan berkumpul untuk melakukan atau menyelesaikan suatu masalah dengan pendekatan yang menekankan pada elemen atau komponen.

2.3 Karakteristik Sistem

Menurut (Jogiyanto 2005)“Dalam sebuah sistem mempunyai karakteristik yang tidak terpisahkan antara satu karakteristik dengan karakteristik yang lain”. Beberapa karakteristik tersebut antara lain:

1. Komponen *(Components*)

   Suatu sistem memiliki sejumlah komponen yang saling berinteraksi, dimana setiap komponen akan membentuk satu kesatuan yang saling bekerja sama. Komponen sistem dapat berupa suatu yang merupakan bagian dari sistem yang lebih besar.

2. Batas Sistem *(Boundary*)

   Merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lain / lingkungan luar, dengan batasan ini kita dapat mengetahui ruang lingkup sistem.

3. Lingkungan Luar Sistem *(Environment)*

   Apapun yang berada di luar batas dari sistem yang mempengaruhi operasi suatu sistem.

4. Penghubung Sistem *(Interface)*

   Merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya. Dengan penghubung ini akan mengalir data-data antara subsistem dimana keluaran *(output)* dari satu subsistem akan menjadi masukan *(input)* untuk subsistem yang lain, sehingga antara satu subsistem dengan subsistem lainnya dapat berintegrasi membentuk satu kesatuan.

5. Masukan *(Input)*

   Merupakan energi yang dimasukkan ke dalam sistem, dimana masukan ini dapat berupa masukan perawatan *(maintenance input)* dan masukan sinyal *(signal input)*. *Maintenance input* adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. *Signal input* adalah energi yang diproses untuk didapatkan keluaran.

6. Keluaran *(Output)*

   Merupakan hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan mampu menjadi masukan baru/informasi yang dibutuhkan.

7. Pengolah *(Process)*

   Suatu sistem pasti mempunyai pengolahan data masukan untuk diolah menjadi sebuah informasi.

8. Sasaran Sistem *(Objectives)*

   Merupakan penentu dari tujuan untuk menentukan masukan yang dibutuhkan dan keluaran yang akan dihasilkan sebuah sistem.

2.4 Metode Pengembangan Sistem

Menurut (Jogiyanto 2005) “untuk menggambarkan kegiatan-kegiatan yang dilaksanakan selama pengembangan suatu sistem digunakan metode pengembangan sistem yaitu *System Development Life Cycle* (SDLC) yang memiliki 4 langkah fundamental” yaitu sebagai berikut :

a). Perencanaan Sistem (*System Planning*)

Pengumpulan data atau fakta yang dapat mendukung suatu sistem untuk dapat dikembangkan atau dibuat sistem baru.

b). Analisis Sistem (*System Analisis*)

Mempelajari masalah-masalah yang timbul dan menentukan kebutuhan pemakai sistem untuk mengidentifikasi pemecahan yang beralasan.

c). Perancangan Sistem (*System Design*)

Perancangan sistem merupakan proses penyiapan spesifikasi yang terperinci untuk pengembangan sistem baru. Dimulai dari spesifikasi output sistem yang diperlukan, mencakup isi, format, volume dan frekuensi laporan-laporan dan dokumen-dokumen juga input sistem dan file.

d). Implementasi Sistem (*System Implementation*)

Pelaksanaan mencakup pelaksanaan alternatif yang dipilih agar sistem siap untuk dioperasikan.

2.5 Pengertian Aplikasi

Menurut (Hendrayudi 2008) menyatakan bahwa "Aplikasi adalah kumpulan perintah program yang dibuat untuk melakukan pekerjaan - pekerjaan tertentu atau khusus".

2.6 Pengertian Tekanan Kompresi Silinder Mesin

Menurut (Daryanto 2003) "Tekanan kompresi silinder mesin adalah tekanan efektif rata-rata yang terjadi di ruang bakar mesin silinder tepat di atas piston". Tekanan kompresi ini juga dibagi dengan 2 definisi, tekanan kompresi motorik dan tekanan kompresi pembakaran. Tekanan kompresi motorik ini adalah tekanan yang sering di ukur oleh mekanik dengan alat compression gauge dengan satuan kPa, psi atau bar. Tekanan motorik akhirnya lebih dikenal dengan tekanan kompresi. Tekanan ini membaca tekanan kompresi di ruang bakar tanpa adanya penyalaan busi. Selanjutnya yang kedua adalah tekanan ruang bakar, tekanan ini dihitung saat mesin menyala atau terjadi proses pembakaran, pengukuran ini tidak menggunakan alat compression gauge lagi, namun memakai sensor pressure yang ditanam di silinder head. Tekanan kompresi pembakaran ini bisa mencapai 10x lipat dari tekanan motorik.

2.7 Pengertian Metode Pohon Keputusan (*Decision Tree*)

Menurut (Hermawati 2013) "Pohon keputusan merupakan representasi sederhana dari teknik klasifikasi untuk sejumlah kelas berhingga, dimana simpul internal maupun simpul akar ditandai dengan nama atribut, rusuk-rusuknya diberi label nilai atribut yang mungkin dan simpul daun ditandai dengan kelas-kelas yang berbeda".

Salah satu algoritma yang digunakan untuk membangun pohon keputusan yang berbasis algoritma induksi pohon keputusan seperti ID3, C45 dan CART adalah algoritma hunt.

Misalkan D_t adalah himpunan *record* pembelajaran yang mencapai suatu simpul *t* dan $y = \{y_1, y_2 ..., y_c\}$ adalah label kelas, prosedur dari algoritma hunt adalah sebagai berikut:

a. Jika semua *record* dalam D_t menempati kelas y_t yang sama, maka *t* adalah simpul daun yang diberi label y_t.
b. Jika D_t adalah himpunan kosong, maka *t* merupakan sebuah simpul daun yang diberi label kelas *default*, y_d.
c. Jika D_t terdiri dari *record* yang menempati lebih dari satu kelas, gunakan sebuah atribut tes untuk membagi data kedalam himpunan bagian yang lebih kecil. Secara rekursif terapkan prosedur ini untuk tiap himpunan bagian.

Pada algoritma Hunt, pohon keputusan tumbuh dalam struktur perulangan dengan membagi *record* pelatihan (*training set*) kedalam bagian yang berurutan. *T* merupakan *record* pelatihan yang berhubungan dengan simpul *t*. Algoritma hunt akan bekerja jika setiap kombinasi pada nilai atribut ada dalam data pelatihan dan setiap kombinasi memiliki label kelas yang unik. Ada beberapa kondisi tambahan untuk menangani beberapa kasus, yaitu:

1. Kemungkinan untuk beberapa simpul anak yang dibuat pada langkah 2 menjadi kosong. Ini dapat terjadi jika tidak ada satupun *record* pelatihan mempunyai nilai kombinasi atribut yang berhubungan dengan beberapa simpul tersebut. Dalam kasus ini, simpul menerangkan simpul daun dengan label kelas yang sama sebagai kelas mayoritas dari *record* pelatihan yang berhubungan dengan simpul orang tua.
2. Pada tahap kedua, jika semua *record* berhubungan dengan C_t mempunyai nilai atribut yang identik (kecuali untuk label kelas), maka tidak mungkin memotong atau membagi *record* tersebut. Pada kasus ini, simpul adalah simpul daun yang mempunyai label kelas yang sama sebagai kelas

mayoritas pada *record* pelatihan yang berhubungan dengan simpul tersebut.

Contoh pembentukan pohon keputusan dengan algoritma Hunt untuk data:

Tabel 2.1 Contoh Pembentukan Pohon Keputusan Dengan Algoritma Hunt

Id	Pembayaran	Status	Pendapatan (x1000)	Kelas
1	Yes	Single	125	No
2	No	Married	100	No
3	No	Single	70	No
4	Yes	Married	120	No
5	No	Divorced	95	Yes
6	No	Married	60	No
7	Yes	Divorced	220	No
8	No	Single	85	Yes
9	No	Married	75	No
10	No	Single	90	Yes

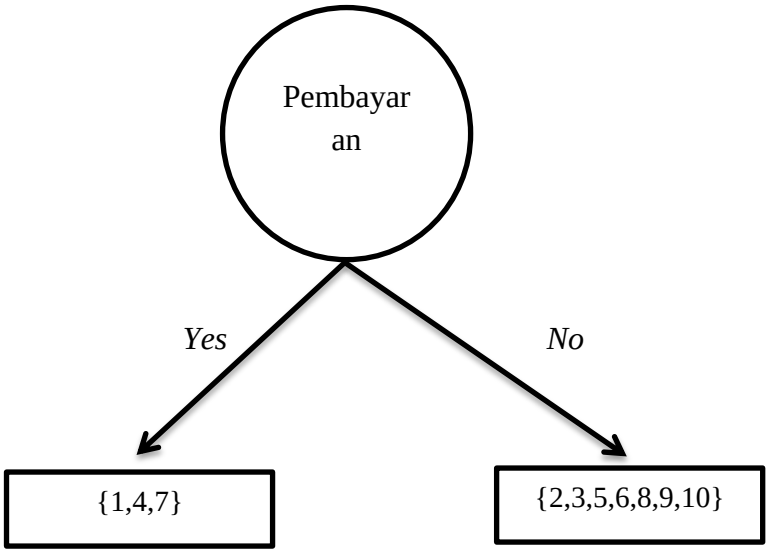
Proses Algoritma Hunt:

1. Mula-mula dibuat simpul D_t beranggotakan seluruh *record* data *training*, D_t ={1,2,3,4,5,6,7,8,9,10}, yang dikatagorikan dalam dua kelas *y*= {*Yes*,*No*}.
2. Cek apakah semua *record* dalam D_t terletak pada satu kelas.
3. D_t tidak terletak pada satu kelas, maka dilakukan proses *splitting* dengan menggunakan salah satu atribut, misalkan atribut **'pembayaran'** . karena atribut 'pembayaran' mempunyai dua nilai atribut yaitu '*Yes*' dan '*No*', maka himpunan D_t dibagi menjadi dua subhimpunan:

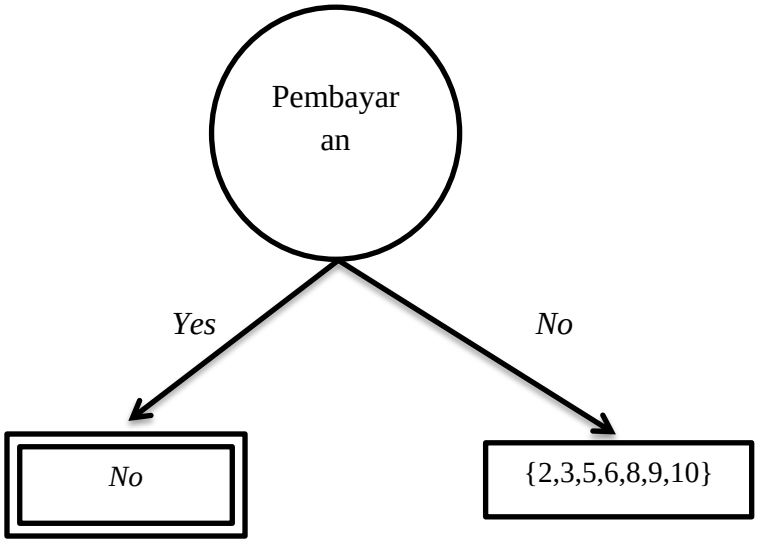
   D_t (Pembayaran='*Yes*')= {1,4,7}

   D_t (Pembayaran='*No*')= {2,3,5,6,8,9,10}.

   Perlu diingat bahwa D_t (pembayaran='*Yes*') ∪ D_t (pembayaran='*No*')= $Dt_{mula\text{-}mula}$, dan *Dt* (pembayaran='*Yes*') ∩ D_t (pembayaran='*No*')= {}



4. Ulangi langkah pengecekan langkah 2 untuk setiap D_t. Misalkan $D_t = D_t$ (pembayaran='*Yes*')={1,4,7}.
5. Karena D_t terletak pada kelas yang sama yaitu kelas '*No*', maka simpul *t* dimana D_t berada menjadi simpul daun dengan label '*No*'.



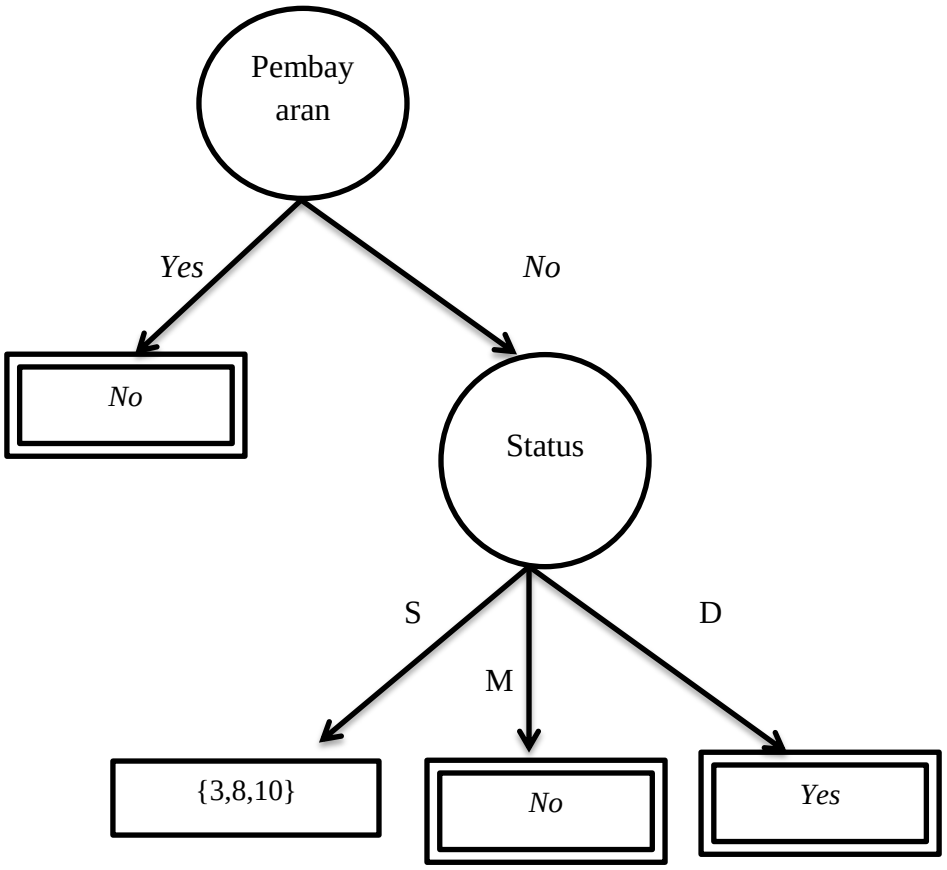
6. Ulangi langkah pengecekan langkah 2 untuk setiap D_t. Misalkan $D_t = D_t$ (pembayaran='*No*')={2,3,5,6,8,9,10}.
7. D_t={2,3,5,6,8,9,10} tidak terletak pada satu kelas, maka dilakukan proses *splitting* dengan menggunakan salah satu atribut sisa, misalkan atribut **'Status'.** Karena atribut 'Status' mempunyai 3 nilai atribut yaitu

'*Single*','*Married*' dan '*Divorced*'. Maka himpunan D_t dibagi menjadi 3 subhimpunan:

D_t (Status= '*Single*')={3,8,10}

D_t (Status= '*Married*')={2,6,9}

D_t (Status= '*Divorced*')={1}

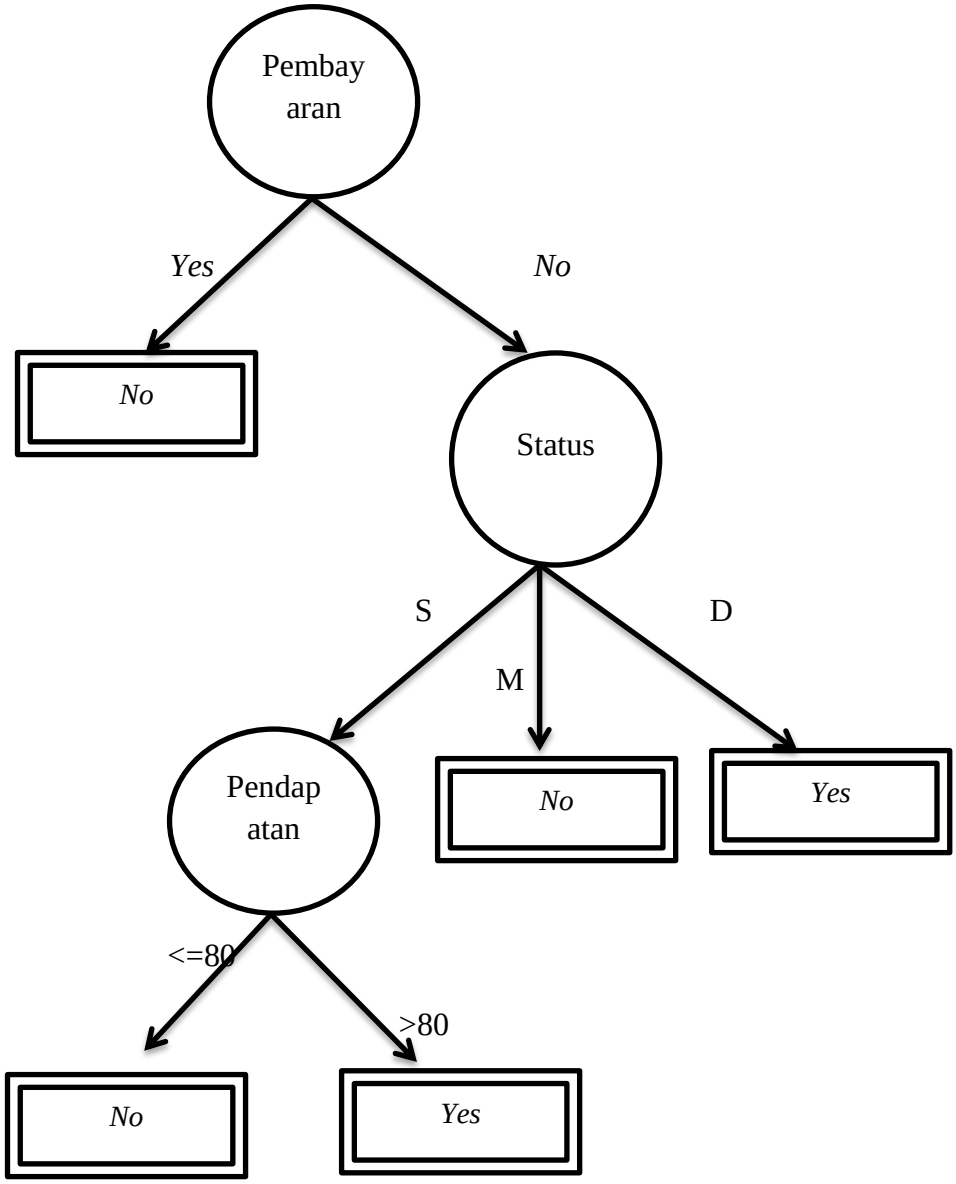


Record {2,6,9} terletak pada kelas yang sama yaitu kelas:'*No*', demikian juga {5} terletak pada kelas '*Yes*', maka keduanya menjadi simpul daun.

8. D_t={3,8,10} tidak terletak pada satu kelas, maka dilakukan proses *splitting*, dengan menggunakan salah satu atribut, misalkan atribut **'Pendapatan'**. Atribut 'Pendapatan' merupakan atribut numerik maka dapat dilakukan proses diskritisasi untuk merubahnya menjadi atribut katagorikal. Misalkan atribut 'Pendapatan' dibagi dalam dua nilai atribut yaitu '<=80' dan ' >=80', maka himpunan D_t dibagi menjadi dua subhimpunan:

D_t (Pendapatan<=80)= {3}

D_t (Pendapatan>80)= {8,10}.



2.8 Pengertian Delphi 7.0

Menurut (Ichwan 2011)"Delphi 7 merupakan bahasa pemrograman yang dikeluarkan pada bulan agustus tahun 2002 oleh *Borland Software Corporation* sebuah perusahaan perangkat lunak komputer yang berkantor pusat di Austin, Texas". Walaupun perkembangan delphi sudah sangat pesat masih banyak pengembang aplikasi yang menggunakan delphi 7, alasannya delphi 7 masih

sangat memadai dan mempunyai kesetabilan yang prima serta kebutuhan perangkat keras yang tidak terlalu tinggi.

2.9 Sistem Pendukung Keputusan

Menurut (Kusrini 2007) ”Sistem pendukung keputusan merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem itu digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi yang tidak terstruktur, dimana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat”.

2.9.1 Dasar-dasar Sistem Pendukung Keputusan

Menurut (Hermawan 2005), Proses pengambilan keputusan melibatkan 4 tahapan, yaitu :

a. Tahap *Intelligence* Dalam tahap ini pengambilan keputusan mempelajari kenyataan yang terjadi sehingga kita bisa mengidentifikasi dan mendefinisikan masalah yang sedang terjadi, biasanya dilakukan analisis berurutan dari sistem ke subsistem pembentuknya. Dari tahap ini didapatkan keluaran berupa dokumen pernyataan masalah.
b. Tahap *Design* Dalam tahap ini pengambilan keputusan menemukan, mengembangkan, dan menganalisis semua pemecahan yang mungkin, yaitu melalui pembuatan model yang bisa mewakili kondisi nyata masalah. Dari tahap ini didapatkan keluaran berupa dokumen alternatif solusi.
c. Tahap *Choice* Dalam tahap ini pengambilan keputusan memilih salah satu alternatif pemecahan yang dibuat pada tahap design yang dipandang sebagai aksi yang paling tepat untuk mengatasi masalah yang sedang dihadapi. Dari tahap ini didapatkan keluaran berupa dokumen solusi dan rencana implentasinya.
d. Tahap *Implementation* Dalam tahap ini pengambilan keputusan menjalankan rangkaian aksi pemecahan yang dipilih di tahap *choice*. Implementasi yang sukses ditandai dengan terjawabnya masalah yang

dihadapi, sementara kegagalan ditandai dengan tetap adanya masalah yang sedang dicoba untuk diatasi. Dari tahap ini didapatkan keluaran berupa laporan pelaksanaan solusi dan hasilnya.

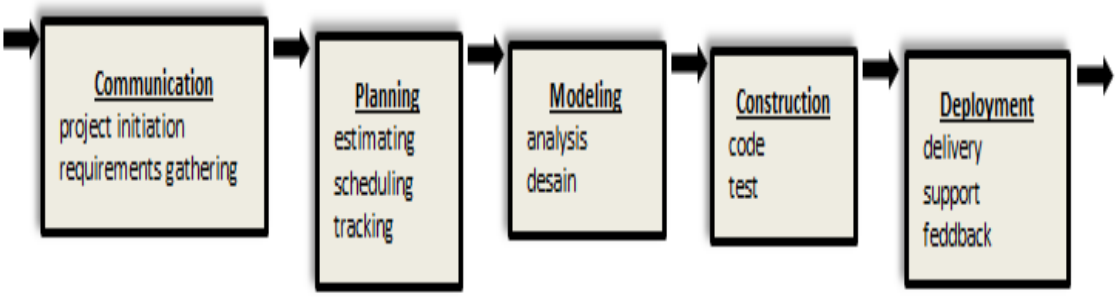
2.9.2 Komponen Sistem Pendukung Keputusan

Menurut (Hermawan 2005), Sistem Pendukung Keputusan terdiri atas tiga komponen penting, yaitu:

1. Manajemen Data Data Management melakukan pengambilan data yang diperlukan baik dari *database* yang berisi data internal maupun *database* yang berisi data eksternal. Jadi, fungsi komponen data ini sebagai pengatur data-data yang diperlukan oleh Sistem Pendukung Keputusan.
2. Manajemen Model Model *Management* melalui Model *Base Management* melakukan interaksi baik dengan *User Interface* untuk mendapatkan perintah maupun Data *Management* untuk mendapatkan data yang akan diolah. Jadi, tujuan dari Model *Management* adalah untuk mengubah data yang ada pada *Database* menjadi informasi yang berguna dalam pengambilan keputusan.
3. Antarmuka Pengguna *User Interface* digunakan untuk berinteraksi antar *user* dengan DSS, baik untuk memasukkan informasi ke sistem maupun menampilkan informasi ke *user*. Karena begitu pentingnya komponen *user interface* bagi suatu sistem DSS, maka harus bisa merancang suatu *user interface* yang bisa mudah dipelajari dan digunakan *user* dan laporan yang bisa digunakan *user* serta pelaporan yang bisa secara mudah dimengerti oleh pengguna. Komponen-komponen tersebut membentuk sistem aplikasi sistem pendukung keputusan yang bisa dikoneksikan ke *intranet* perusahaan, *ekstranet* atau *internet*.

2.10 Model Waterfall

Menurut (Pressman 2010) ”model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Berikut ini ada dua gambaran dari *waterfall* model”. Fase-fase dalam model *waterfall* menurut referensi Pressman:



Gambar 2.1 *Waterfall Pressman*

1. *Communication*

Langkah ini merupakan analisis terhadap kebutuhan *software*, dan tahap untuk mengadakan pengumpulan data dengan melakukan pertemuan dengan *customer*, maupun mengumpulkan data-data tambahan baik yang ada di jurnal, artikel, maupun dari *internet.*

2. *Planning*

Proses *planning* merupakan lanjutan dari proses *communication (analysis requirement).* Tahapan ini akan menghasilkan dokumen *user requirement* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan *software*, termasuk rencana yang akan dilakukan.

3. *Modeling*

Proses *modeling* ini akan menerjemahkan syarat kebutuhan ke sebuah perancangan *software* yang dapat diperkirakan sebelum dibuat *coding*. Proses ini berfokus pada rancangan struktur data, arsitektur *software*, *representasi interface*, dan detail (algoritma) prosedural. Tahapan ini akan menghasilkan dokumen yang disebut *software requirement.*

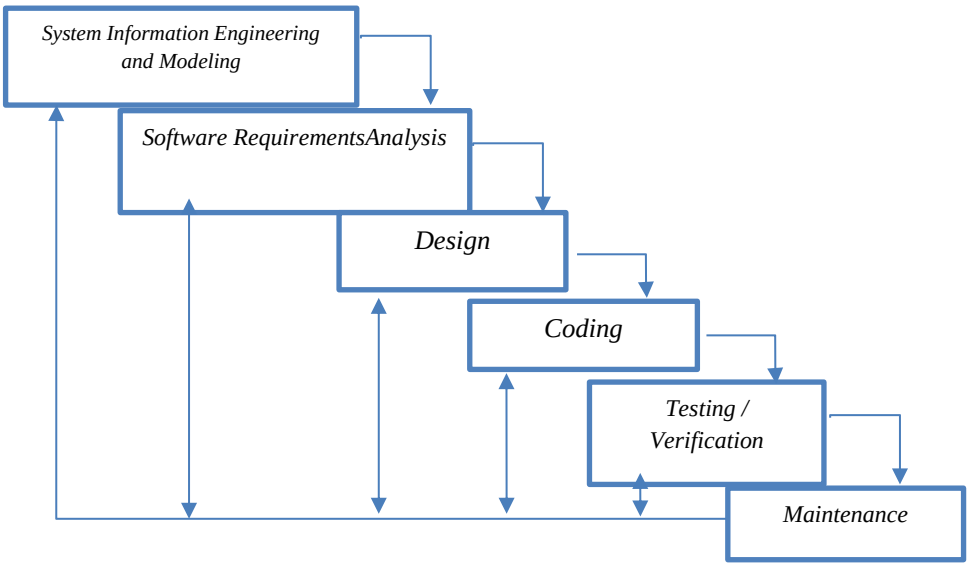
4. *Construction*

Construction merupakan proses membuat kode. *Coding* atau pengkodean merupakan penerjemahan desain dalam bahasa yang bisa dikenali oleh komputer. *Programmer* akan menerjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu *software*, artinya penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap sistem tersebut untuk kemudian bisa diperbaiki.

5. *Deployment*

Tahapan ini bisa dikatakan final dalam pembuatan sebuah *software* atau sistem. Setelah melakukan analisis, desain dan pengkodean maka sistem yang sudah jadi akan digunakan oleh *user*. Kemudian *software* yang telah dibuat harus dilakukan pemeliharaan secara berkala.

Sedangkan fase-fase model *waterfall* menurut referensi Sommerfille:



Gambar 2.2 *Waterfall Sommerfille*

1. *Requirements Analysis and Definition*

Mengumpulkan kebutuhan secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh *software* yang akan dibangun. Hal ini sangat penting, mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware, database*, dsb. Tahap ini sering disebut dengan *Project Definition.*

2. *System and Software Design*

Proses pencarian kebutuhan diintensifkan dan difokuskan pada *software*. Untuk mengetahui sifat dari program yang akan dibuat, maka para *software engineer* harus mengerti tentang domain informasi dari *software*, misalnya fungsi yang dibutuhkan, *user interface*, dsb. Dari dua aktivitas tersebut (pencarian kebutuhan sistem dan *software*) harus didokumentasikan dan ditunjukkan kepada *user*. Proses *software design* untuk mengubah kebutuhan-kebutuhan di atas menjadi representasi ke dalam bentuk "*blueprint*" *software* sebelum *coding* dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya. Seperti dua aktivitas sebelumnya, maka proses ini juga harus didokumentasikan sebagai konfigurasi dari *software*.

3. *Implementation and Unit Testing*

Desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji baik secara unit.

4. *Integration and System Testing*

Untuk dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu ke dalam bahasa pemrograman melalui proses *coding.* Tahap ini merupakan implementasi dari tahap *design* yang secara teknis nantinya dikerjakan oleh *programmer*. Penyatuan unit-unit program kemudian diuji secara keseluruhan *(system testing)*.

5. *Operation and Maintenance*

Sesuatu yang dibuat haruslah diujicobakan. Demikian juga dengan *software*. Semua fungsi-fungsi *software* harus diujicobakan, agar *software* bebas dari *error*, dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya. Pemeliharaan suatu *software* diperlukan, termasuk di dalamnya adalah pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu. Ketika dijalankan mungkin saja masih ada *error* kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebut. Pengembangan diperlukan ketika adanya perubahan dari *eksternal* perusahaan seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

Kelebihan dari model ini adalah selain karena pengaplikasian menggunakan model ini mudah, kelebihan dari model ini adalah ketika semua kebutuhan sistem dapat didefinisikan secara utuh, eksplisit, dan benar di awal proyek, maka *Software Engineering (SE)* dapat berjalan dengan baik dan tanpa masalah. Meskipun seringkali kebutuhan sistem tidak dapat didefinisikan se-eksplisit yang diinginkan, tetapi paling tidak, *problem* pada kebutuhan sistem di awal proyek lebih ekonomis dalam hal uang (lebih murah), usaha, dan waktu yang terbuang lebih sedikit jika dibandingkan *problem* yang muncul pada tahap-tahap selanjutnya. Kekurangan yang utama dari model ini adalah kesulitan dalam mengakomodasi perubahan setelah proses dijalani. Fase sebelumnya harus lengkap dan selesai sebelum mengerjakan fase berikutnya. Masalah dengan *waterfall* :

1. Perubahan sulit dilakukan karena sifatnya yang tidak fleksibel.
2. Dengan sifat yang tidak fleksibel, model ini cocok ketika kebutuhan dikumpulkan secara lengkap sehingga perubahan bisa ditekan sekecil mungkin. Tapi pada kenyataannya jarang sekali konsumen/pengguna yang bisa memberikan kebutuhan secara lengkap, perubahan kebutuhan adalah sesuatu yang wajar terjadi.
3. *Waterfall* pada umumnya digunakan untuk rekayasa sistem yang besar yaitu dengan proyek yang dikerjakan di beberapa tempat berbeda, dan dibagi menjadi beberapa bagian sub-proyek.

2.11 UML (*Unified Markup Languange*)

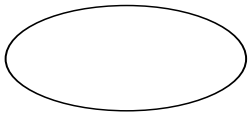
Menurut (Rosa & Salahudin 2014) menyatakan "UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodelogi dalam mengembangkan sistem berorientasi object dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum pada dunia industri perangkat lunak dan pengembangan software".

Alat bantu yang dipergunakan dalam perancangan berorientasi objek berbasis UML adalah sebagai berikut :

2.11.1 *Use Case Diagram*

Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem yang dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih *actor* dengan sistem yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang dibuat dan siapa saja yang berhak mengunakan fungsi-fungsi tersebut. Berikut simbol yang digunakan pada *use case diagram* :

Tabel 2.2 Simbol *Use Case Diagram*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan *system* sebagai unit-unit yang bertukar pesan antar unit dengan *actor*, biasanya dinyatakan dengan mengunakan kata kerja diawal nama *use case*.
 Actor	Aktor adalah orang lain atau *system* lain yang mengaktifkan fungsi dari target *system*. Untuk megidentifikasikan *actor*, harus ditentukan pembagian kerja dan tugas-tugas yang berkaitan dengan peran pada target *system*. Orang atau *system* biasa muncul dalam beberapa peran.

Lanjutan Tabel 2.2 Simbol *Use Case Diagram*

	Ososiasi antara *actor* dan *use case* yang mengunakan panah terbuka untuk mengindikasikan bila *actor* berinteraksi secara pasif dengan *system*.
	Ososiasi antara *actor* dan *use case* yang mengunakan garis untuk mengindikasikan siapa atau apa yang meminta interaksi langsung bukan mengindikasikan aliran data.
 <<include>>	*Include* merupakan pemangilan *use case* oleh *use case* lainnya, contohnya pemanggilan sebuah fungsi program.
 <<extend>>	Perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

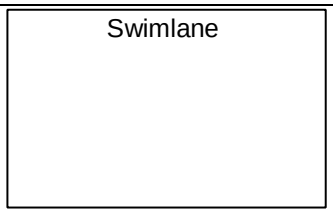
2.11.2 *Activity Diagram*

Secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas baik proses bisnis maupun *use case*. *Activity diagram* dapat juga digunakan untuk memodelkan *action* yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari *action* tersebut**.** Simbol-simbol yang digunakan dalam *activity diagram* yaitu :

Tabel 2.3 Simbol *Activity Diagram*

Gambar	Keterangan
	Mengambarkan suatu proses/kegiatan sistem
	Menunjukan eksekusi dari suatu aksi
	*Start Point* merupakan awak dari suatu aktivitas

Lanjutan Tabel 2.3 Simbol *Activity Diagram*

	*End point*, akhir aktivitas.
	Fork/Rake Node Satu aliran atau beberapa aliran yang pada tahap tertentu berubah menjadi satu atau beberapa aliran.
	*Swinlane*, pembagian *activity diagram* untuk menunjukan siapa melakukan apa.

2.11.3 *Class Diagram*

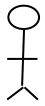
Class diagram merupakan hubungan antara kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sitem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan.

Class diagram secara khas meliputi : kelas (*class*), relasi, *Associations*, *Generalizion* dan *Agregation*, Atribut, operasi, dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau *object*.

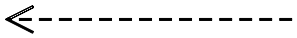
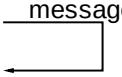
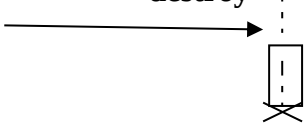
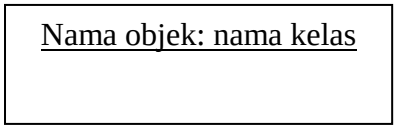
2.11.4 *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeksripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel 2.4 Simbol *Sequence Diagram*

Gambar	Keterangan
	Orang, proses atau *system* lain yang berinteraksi dengan sistem.

Lanjutan Tabel 2.4 Simbol *Sequence Diagram*

1: Masukan 	Menyatakan bahwa suatu objek mengirimkan data atau masukan atau informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek.
1: Keluaran 	*Message return*, simbol membalas pesan.
1: [condition] message name 	*Recursive*, menggambarkan pengiriman pesan untuk dirinya sendiri.
<<create>>	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah objek yang di buat.
<<destroy>> 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*.
Garis Hidup / Lifeline	Menyatakan kehidupan suatu objek.
	Menyatakan objek yang berinteraksi pesan.