BAB II

LANDASAN TEORI

2.1 Sistem

Menurut Badri (2007) Sistem terdiri dari subsistem yang berhubungan dengan prosedur yang membantu pancapain tujuan. Pada saat prosedur diperlukan untuk melengkapi beberapa proses pekerjaan, maka metode berisi tentang aktivitas operasional atau teknis yang akan menjelaskanya.

2.2 Proses Produksi

Proses produksi yaitu suatu kegiatan perbaikan terus-menerus (*continuos improvment*), yang dimulai dari sederet siklus sejak adanya ide-ide untuk menghasilkan suatu produk, pengembangan produk, proses produksi, sampai distribusi kepada konsumen (Gaspersz, 2004).

Secara umum, proses produksi dibedakan menjadi dua jenis, yaitu proses produksi yang terus-menerus (*countinous processes*) dan proses produksi yang terputus-putus (*intermittent processes*). Perbedaan dari kedua proses produksi tersebut adalah berdasarkan pada panjang tidaknya waktu persiapan untuk mengatur (*set up*) peralatan produksi yang digunakan untuk memproduksi suatu produk atau beberapa produk tanpa mengalami perubahan. Pada proses produksi yang terus-menerus, perusahaan atau pabrik menggunakan mesin-mesin yang dipersiapkan (*set up*) dalam jangka waktu yang lama dan tanpa mengalami perubahan. Sedangkan untuk proses produksi yang terputus-putus menggunakan mesin-mesin yang dipersiapkan dalam jangka waktu yang pendek, dan kemudian akan dirubah atau dipersiapkan kembali untuk memproduksi produk lain. Selain

countinous processes dan *intermittent processes* juga terdapat proses produksi yang bersifat proyek, di mana kegiatan produksi dilakukan pada tempat dan waktu yang berbeda – beda, sehingga peralatan produksi yang digunakan ditempatkan di lokasi di mana proyek tersebut dilaksanakan dan pada saat yang direncanakan.

2.2.1 Pentingnya Perencanaan Produksi

Pentingnya perencanaan produksi (Tersine dalam Pangestu, 2005) dimaksudkan untuk mengadakan persiapan produksi, sehingga proses produksi dapat berjalan dengan lancar. Persiapan tersebut meliputi persiapan bahan baku, tenaga kerja, mesin-mesin dan peralatan lainnya yang dibutuhkan untuk memproduksi barang.

Dengan adanya perencanaan produksi, maka kita dapat mengatehui beberapa jumlah bahan baku yang dibutuhkan dan beraa lama waktu penyelesaian suatu produk sehingga tidak terjadi keterlambatan serta proses produksi berjalan dengan lancar.

Adapun tujuan dari perencanaan prosuksi adalah:

1. Menentukan jumlah persediaan bahan baku yang optimal.
2. Dapaat memenuhi permintaan pelanggan dengan tepat.
3. Menentukan persiapan-persiapan yang dibutuhkan untuk produksi secara tepat.

2.3 Penjadwalan Produksi

Menurut Arman, dkk (2008), penjadwalan produksi dapat didefinisikan sebagai proses pengalokasian sumber daya dan mesin yang ada untuk menyelesaikan semua pekerjaan dengan mempertimbangkan batasan-batasan yang

ada. Pada saat merencanakan suatu jadwal produksi, ketersediaan sumber daya yang dimiliki harus dipertimbangkan dengan baik.

Menurut Arman, dkk (2008), tujuan dari penjadwalan produksi adalah:

1. Meningkatkan penggunaan sumber daya atau mengurangi waktu tunggunya, sehingga total waktu proses dapat berkurang dan produktivitasnya dapat meningkat.
2. Mengurangi persediaan barang setengah jadi atau mengurangi sejumlah pekerjaan yang menunggu dalam antrian ketika sumber daya yang ada masih mengerjakan tugas yang lain.
3. Mengurangi beberapa keterlambatan pada pekerjaan yang mempunyai batas waktu penyelesaian sehingga akan meminimaliasi biaya keterlambatan.
4. Membantu pengambilan keputusan mengenai perencanaan kapasitas pabrik dan jenis kapasitas yang dibutuhkan sehingga penambahan biaya dapat dihindarkan.

Penjadwalan produksi memiliki beberapa metode dalam penyelesaiannya, adapun metode-metode tersebut yang digunakan antara lain:

1. Metode *Shortest Processing Time* (SPT), metode ini memprioritaskan operasi atau pekerjaan yang waktu prosesnya terpendek. Metode ini meminimalkan *work in proces*, rata-rata keterlambatan (*mean latenes)* dan waktu penyelesaian rata-rata (*mean flow time)* produk.
2. Metode *Longest Processing Time* (LPT), metode ini memprioritaskan operasi atau pekerjaan yang mempunyai waktu proses yang terlama dulu yang akan dikerjakan terlebih dahulu.

3. Metode *Earliest Due Date* (EDD), metode ini memprioritaskan operasi atau pekerjaan yang batas penyelesaiannya (*Due Date*) terpendek. Metode ini berjalan baik pada pekerjaan yang waktu prosesnya relatif sama.
4. FCFS (*First Come First Server*), metode yang memprioritaskan operasi atau pekerjaan yang datang terlebih dahulu. Pekerjaan atau proses yang datang akan langsung diproses terlebih dahulu. Metode ini sangat cocok untuk perusahaan atau organisasi yang pelanggannya lebih mementingkan waktu pelayanan.
5. Algoritma Hodgson, digunakan untuk meminimalkan jumblah pekerjaan yang terlambat.

Adapun kriteria-kriteria yang dapat digunakan sebagai dasar pemilihan metode penjadwalan yang sesuai antara lain (Arman, dkk, 2008):

1. *Mean Flow Time*: rata-rata waktu tinggal pekerjaan dalam sistem.
2. *Makespam*: waktu penyelesaian semua pekerjaan.
3. *Tardiness*: keterlambatan.
4. *Maximum Tardiness*: keterlambatan maksimum.
5. *Mean Tardiness*: rata-rata waktu keterlambatan.
6. *Number of Tardy Job*: jumlah pekerjaan yang terlambat.

2.4 Istilah-istilah dalam Penjadwalan Produksi

Beberapa istilah umum yang digunakan dalam penjadwalan produksi antara lain:

1. *Processing time* (waktu proses), merupakan perkiraan waktu penyelesaiaan suatu pekerjaan atau waktu yang dibutuhkan untuk melakukan suatu proses tertentu. Waktu proses ini disimbolkan dengan T_i.

2. *Due Date* (batas waktu), merupakan batas waktu yang ditetapkan untuk suatu pekerjaan. Jika pekerjaan diselesaikan dengan waktu lebih lama dari *due date*, maka pekerjaan dianggap terlambat. Diassumsikan bahwa beberapa jenis penalty akan terjadi jika pekerjaan terlambat. *Due date* dinotasikan sebagai d_i.
3. *Lateness* (keterlambatan), merupakan Selisih antara waktu penyelesaian pekerjaan dengan batas waktunya (*due date*). Suatu pekerjaan dapat memiliki kelambatan positif, dalam hal pekerjaan selesai setelah batas waktunya atau kelambatan negatif, dimana pekerjaan selesai sebelum batas waktu yang ditetapkan. *Lateness* dinotasikan sebagai L_i.
4. *Tardiness* (ukuran keterlambatan), merupakan kuran kelambatan positif. Jika pekerjaan selesai lebih awal dari batas waktu, pekerjaan ini akan memiliki *Lateness* negatif tetapi *Tardiness* positif. *Tardiness* dinotasikan sebagai T_i dimana T_i ialah maksimum $(0,L_i)$.
5. *Slack* (kelonggaran), merupakan ukuran selisih antara waktu yang tersisa antara saat selesainya pekerjaan dengan batas waktu yang ditetapkan. *Slack* dinotasikan sebagai $Sl_i = d_i - t_i$.
6. *Completion Time* (waktu penyelesaian), merupakan rentang antara awal pekerjaan pada pekerjaan pertama, dimana pekerjaan dinotasikan sebagai t = 0 dan saat dimana pekerjaan ke-i diselesaikan. Rentang ini dinotasikan sebagai C_i.
7. *Flow Time* (waktu alir), merupakan, Rentang waktu antara saat suatu pekerjaan dapat dimulai dan saat pekerjaan selesai dikerjakan. Sehingga *flow*

time sama dengan waktu pemrosesan ditambah dengan waktu menunggu sebelum diproses. Waktu alir dinotasikan sebagai F_i.

8. *Idle Time* (waktu tunggu), merupakan durasi waktu di saat sebuah mesin dalam kondisi statis. Dengan kata lain mesin itu hidup atau aktif, tetapi tidak dapat dipakai untuk bekerja.

2.5 Metode *Earliest Due Date* (EDD)

Menurut Bedworth dalam Pangestu (2005) Metode *Earliest Due Date* menjelaskan bahwa pengurutan pekerjaan berdasarkan batas waktu (*Due Date*) tercepat. Pekerjaan dengan saat jatuh tempo paling awal harus dijadwalkan terlebih dahulu daripada pekerjaan dengan saat jatuh tempo paling akhir. Metode ini dapat digunakan untuk penjadwalan pada satu mesin (*single machine*) maupun untuk penjadwalan pada beberapa mesin (*pararel machine*). Metode ini bertujuan untuk meminimasi kelambatan maksimum (*Maximum Lateness*) atau meminimasi ukuran kelambatan maksimum (*Maksimum Tardiness*) suatu pekerjaan.

Parameter-parameter yang diperlukan dalam penjadwalan dengan metode *Earliest Due Date* ini adalah waktu pemrosesan dan *due date* tiap pekerjaan. Langkah-langkah penggunaan petode ini antara lain:

Langkah 1: Urutkan pekerjaan berdasarkan tanggal jatuh tempo terdekat.

Langkah 2: Ambil pekerjaan satu persatu dari urutan berdasarkan tanggal jatuh tempo itu lalu jadwalkan pada mesin dengan beban yang paling minimum. Jika ada 2 mesin atau lebih yang memiliki beban paling minimum, jadwalkan pekerjaan pada salah satu mesin secara random.

Sebagai contoh, sebuah perusahaan mendapatkan order 10 buah pesanan dengan data sebagai berikut:

Tabel 2.1 Contoh Data Pesanan

pekerjaan	waktu proses t_i (jam)	due date d_i (jam)
1	3	10
2	3	15
3	3	20
4	4	10
5	2	10
6	5	15
7	3	8
8	3	40
9	2	50
10	4	35

Urutkan pekerjaan berdasarkan tanggal jatuh tempo terdekat.

Tabel 2.2 Perhitungan EDD

pekerjaan	waktu proses t_i (jam)	Completion Time c_i (jam)	due date d_i (jam)	lateness L_i
7	3	3	8	3-8 = -5
1	3	3+3 = 6	10	6-10 = -4
4	4	6+4 = 10	10	10-10 = 0
5	2	10+2 = 12	10	12-10 = 2
2	3	12+3 = 15	15	15-15 = 0

pekerjaan	waktu proses t_i (jam)	Completion Time c_i (jam)	due date d_i (jam)	lateness L_i
6	5	15+5 = 20	15	20-15 = 5
3	3	20+3 = 23	20	23-20 = 3
10	4	23+4 = 27	35	27-35 = 8
8	3	27+3 = 30	40	30-40 = -10
9	2	30+3 = 32	35	32-35 = -3

2.6 Perbandingan Metode

Perbandingan ini dimaksudkan untuk mengetahui apakah metode EDD menghasilkan keterlambatan maksimum lebih kecil dari pada metode-metode yang lain atau tidak. Perbandinga tersebut dapat dilihat pada tabel 2.3, tabel 2.4 dan tabel 2.5.

Tabel 2.3 Metode *Shortest Processing Time* (SPT)

Pekerjaan	Waktu Proses ti (jam)	Completion Time c_i (jam)	Due Date d_i (jam)	lateness L_i
5	2	2	10	-8
9	2	4	50	-46
1	3	7	10	-3
2	3	10	15	-5
3	3	13	20	-7
7	3	16	8	8
8	3	19	15	4
4	4	23	10	13
10	4	27	35	-8
6	5	32	15	17

Tabel 2.4 Metode *Longest Processing Time* (LPT)

Pekerjaan	Waktu Proses ti (jam)	Completion Time c_i (jam)	Due Date di (jam)	lateness L_i
6	5	5	15	-10
4	4	9	10	-1
10	4	13	35	-22

Pekerjaan	Waktu Proses ti (jam)	Completion Time c_i (jam)	Due Date di (jam)	lateness L_i
1	3	16	10	6
2	3	19	15	4
3	3	22	20	2
7	3	25	8	17
8	3	28	40	-12
5	2	30	10	20
9	2	32	50	-18

Tabel 2.5 Metode FCFS (*First Come First Server*)

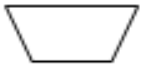
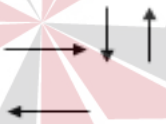
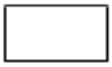
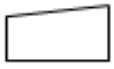
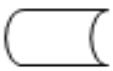
Pekerjaan	Waktu Proses ti (jam)	Completion Time c_i (jam)	Due Date di (jam)	lateness L_i
1	3	3	10	-7
2	3	6	15	-9
3	3	9	20	-11
4	4	13	10	3
5	2	15	10	5
6	5	20	15	5
7	3	23	8	15
8	3	26	40	-14
9	2	28	50	-22
10	4	32	35	-3

Dari hasil perhitungan 3 metode diatas, dapat diketahui bahwa metode EDD menghasilkan keterlambatan maksismum paling kecil dengan keterlambatan sebesar 8 jam, dibandingkan dengan metode SPT dengan keterlambatan sebesar 17 jam, LPT dengan keterlambatan sebesar 20 jam dan FCFS dengan keterlambatan sebesar 15 jam.

2.7 Diagram Alir Dokumen (*Document Flowchart*)

Menurut Jogiyanto (2005), diagram alir dokumen atau paperwork flowchart merupakan diagram alir yang menunjukkan arus laporan dan formulir beserta tembusannya. Berdasarkan pengertian di atas dapat disimpulkan bahwa

diagram alir dokumen adalah diagram yang menggambarkan aliran seluruh dokumen. Diagram alir dokumen ini menggunakan simbol-simbol yang sama dengan diagram alir sistem. Diagram alir dokumen digambar dengan menggunakan simbol-simbol yang ada pada gambar berikut (Jogiyanto, 2005):

Simbol	Keterangan
	**Dokumen** Menunjukan dokumen berupa dokumen input dan output pada proses manual dan proses berbasis komputer
	**Proses Manual** Menunjukan proses yang dilakukan secara manual.
	**Penyimpanan Magnetik** Menunjukan media penyimpanan data/informasi file pada proses berbasis komputer. File dapat disimpan pada harddisk, disket, CD dan lain-lain.
	**Arah Alir Dokumen** Menunjukan arah aliran dokumen antar bagian yang terkait pada suatu sistem. Bisa dari sistem keluar ataupun dari luar ke sistem dan antar bagian diluar system.
	**Penghubung** Menunjukan alir dokumen yang terputus atau terpisah pada halaman alir dokumen yang sama.
	**Proses Komputer** Menunjukan proses yang dilakukan secara komputerisasi.
	**Pengarsipan** Menunjukan simpanan data non-komputer/informasi file pada proses manual. Dokumen dapat disimpan pada lemari, arsip, map file dan lain-lain.
	**Input Keyboard** Menunjukan *input* yang dimasukan melalui *keyboard*.
	**Penyimpanan Manual** Menunjukan media penyimpanan data/informasi secara manual.

Gambar 2.1 Simbol-simbol *Document Flowchart*

2.8 Diagram Alir Sistem (*System Flowchart*)

Diagram alir sistem merupakan diagram alir yang menggambarkan suatu sistem peralatan komputer yang digunakan untuk mengolah data dan menghubungkan antar peralatan tersebut (Oetomo, 2002). Diagram alir sistem ini tidak digunakan untuk menggambarkan langkah-langkah dalam memecahkan masalah tetapi hanya menggambarkan prosedur pada sistem yang dibentuk. Diagram alir sistem digambar dengan menggunakan simbol-simbol tertentu. Ada dua jenis simbol yang digunakan untuk menggambar diagram alir sistem, yaitu:

1. *Flow Direction Symbols*

   Flow direction symbols digunakan untuk menghubungkan antara satu simbol dengan simbol lainnya.

2. *Processing Symbols*

   Processing symbols merupakan simbol yang menunjukkan jenis operasi pengolahan data dalam suatu proses. Simbol-simbol tersebut dijelaskan pada tabel di bawah ini:

2.9 Diagram Konteks (*Context Diagram*)

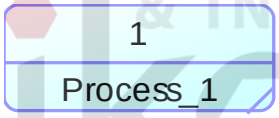
Diagram konteks merupakan sebuah model proses yang digunakan untuk mendokumentasikan ruang lingkup dari sebuah sistem (Whitten, 2004). Menurut Oetomo (2002), terdapat beberapa hal yang perlu diperhatikan dalam membuat diagram konteks, diantaranya:

1. Kelompok pemakai, baik internal maupun eksternal perusahaan.
2. Identifikasi kejadian-kejadian yang mungkin terjadi dalam penggunaan sistem.

3. Arah anak panah yang menunjukkan aliran data.
4. Setiap kejadian digambarkan dalam bentuk yang sederhana dan mudah dipahami oleh pembuat sistem.

Suatu diagram konteks hanya mengandung satu proses saja, biasanya diberi nomor proses 0. Proses ini mewakili proses dari seluruh sistem dengan dunia luarnya. Simbol-simbol yang digunakan dalam membuat diagram konteks digambarkan pada tabel di bawah ini:

Tabel 2.6 Simbol-simbol *Context Diagram*

No	Nama	Simbol	Fungsi
1.	*External Entity*		Simbol ini digunakan untuk berkomunikasi dengan sistem aliran data.
2.	*Process*		Simbol ini berfungsi untuk mewakili suatu aktifitas yang ada pada sistem.
3.	*Flow* (Aliran data)		Simbol ini digunakan untuk menunjukkan arah dari aliran data.

2.10 *Data Flow Diagram* (DFD)

Menurut Whitten (2004), Data Flow Diagram (DFD) merupakan alat yang menggambarkan aliran data melalui sistem. Dalam pembuatan DFD, terdapat beberapa tingkatan yang bertujuan untuk menghindari aliran data yang rumit.

Tingkatan tersebut dimulai dari tingkatan tertinggi ke bentuk yang lebih rinci. Tingkatan DFD terdiri atas:

1. Diagram Konteks (*Context Diagram*)

Diagram konteks merupakan sebuah model proses yang digunakan untuk mendokumentasikan ruang lingkup dari sebuah sistem (Whitten, 2004).

2. Diagram level 0

Diagram level 0 merupakan diagram aliran data yang menggambarkan sebuah event konteks. Diagram ini menunjukkan interaksi antara input, output, dan data store pada setiap proses yang ada.

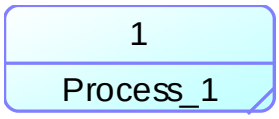
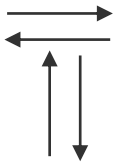
3. Diagram rinci

Diagram rinci menggambarkan rincian dari proses yang ada pada tingkatan sebelumnya. Diagram ini merupakan diagram dengan tingkatan paling rendah dan tidak dapat diuraikan lagi.

DFD terdiri atas empat simbol. Simbol-simbol tersebut digambarkan pada tabel di bawah ini:

Tabel 2.7 Simbol-simbol DFD

No	Nama	Simbol	Fungsi
1.	*External Entity*		External entity merupakan kesatuan di lingkungan luar sistem yang dapat berupa orang, organisasi, atau sistem lainnya yang akan memberikan input ataupun menerima output.

No	Nama	Simbol	Fungsi
2.	*Process*		Proses adalah kegiatan yang dilakukan oleh orang atau komputer dari arus data yang masuk untuk menghasilkan arus data yang keluar.
3.	*Data Flow* (Aliran data)		Data flow atau aliran data yang mengalir diantara proses. Aliran data dapat digambarkan dari bawah ke atas, kiri ke kanan, maupun sebaliknya.
4.	*Data Store*		Data store merupakan tempat penyimpanan data yang berupa file maupun database di dalam sistem komputer.

Setiap simbol memiliki aturan tersendiri dalam penggunaannya. Aturan-aturan tersebut antara lain:

1. *External Entity* (Entitas Luar)

   Aturan penggunaan untuk external entity antara lain:

a. Data harus bergerak melalui proses, selama data tersebut berhubungan dengan sistem. Jika data tidak berhubungan dengan proses, maka aliran data tidak perlu ditampilkan pada DFD.
b. Entitas luar diberi label dengan sebuah frase kata benda.

2. Process

Aturan penggunaan untuk *Process* antara lain:

a. Sebuah proses tidak hanya memiliki output. Jika sebuah objek hanya memiliki output, maka objek tersebut adalah source.
b. Sebuah proses tidak hanya memiliki input. Jika sebuah objek hanya memiliki sebuah input, maka objek tersebut adalah entitas luar.
c. Sebuah proses diberi label dengan sebuah frase kata kerja.

3. *Data Flow* (Aliran Data)

Aturan penggunaan untuk *Data Flow* (Aliran Data) antara lain:

a. Sebuah aliran data hanya menggunakan satu arah antar simbol.
b. Sebuah cabang pada aliran data memiliki arti data yang sama dari satu lokasi menuju ke satu atau lebih proses, tempat penyimpanan data, serta entitas luar.
c. Sebuah aliran data tidak dapat bergerak ke proses asalnya sehingga membutuhkan proses lain untuk menangani, menghasilkan, dan mengembalikan aliran data ke proses asal.
d. Aliran data atau data flow diberi label dengan frase kata benda.

4. Data Store

Aturan penggunaan untuk *Data Store* antara lain:

a. Data harus bergerak melalui proses dimana data diterima melalui suatu source untuk disimpan di data store.

b. Data tidak dapat bergerak langsung dari data source menuju external entity.

c. *Data store* diberi label dengan frase kata benda.

2.11 *Entity Relationship Diagram* (ERD)

Pengertian *Entity Relation Diagram* (ERD) menurut Jogiyanto (2001) adalah suatu komponen himpunan entitas dan relasi yang dilengkapi dengan atribut yang mempresentasikan seluruh fakta. ERD digunakan untuk menggambarkan model hubungan data dalam sistem yang di dalamnya terdapat hubungan entitas berserta atribut relasinya serta mendokumentasikan kebutuhan sistem untuk pemrosesan data. ERD memiliki 4 jenis objek, antara lain:

1. *Entity*

Entitas adalah kelompok orang, tempat, objek, kejadian atau konsep tentang apa yang diperlukan untuk menyimpan data (Whitten, 2004). Setiap entitas yang dibuat memiliki tipe untuk mengidentifikasi apakah entitas tersebut bergantung dengan entitas lainnya atau tidak. Tipe entitas merupakan kumpulan objek yang memiliki kesamaan properti yang teridentifikasi oleh perusahaan dan memiliki keberadaan yang independen (Connolly & Begg, 2002). Tipe entitas terdiri atas dua jenis, yaitu:

a. *Strong Entity*

Strong Entity adalah tipe entitas yang tidak bergantung pada keberadaan jenis entitas lainnya. Suatu entitas dikatakan kuat apabila tidak tergantung pada entitas lainnya.

b. *Weak Entity*

Weak Entity adalah tipe entitas yang bergantung pada keberadaan jenis entitas lain yang saling berhubungan. Karakteristik weak entity terletak pada

entitas occurrence yang tidak dapat teridentifikasi secara unik. Entitas occurrence adalah sebuah objek yang secara unik dapat teridentifikasi dengan tipe entitas (Connolly & Begg, 2002).

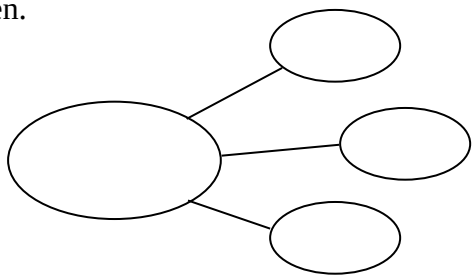
2. *Attribute*

Menurut Connolly dan Begg (2002) atribut adalah deskripsi data yang mengidentifikasikan dan membedakan suatu entitas dengan entitas lainnya. Setiap atribut memiliki domain untuk mendefinisikan nilai-nilai potensial yang dapat menguatkan atribut. Atribut domain adalah kumpulan nilai-nilai yang diperbolehkan untuk satu atau lebih atribut (Connolly & Begg, 2002). Atribut dapat dibedakan menjadi 5 jenis, yaitu:

a. *Simple Attribute* adalah atribut yang terdiri dari komponen tunggal. Simpel atribut tidak dapat dibagi menjadi komponen yang lebih kecil.

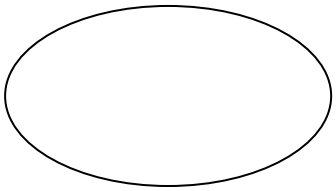
Gambar 2.2 Simbol *Simple Attribute*

b. *Composite Attribute* adalah atribut yang terdiri dari beberapa komponen yang bersifat independen.



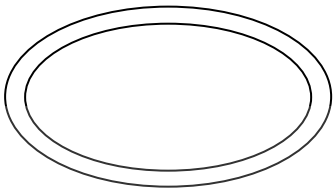
Gambar 2.3 Simbol *Composite Attribute*

c. *Single-value Attribute* adalah atribut yang memegang nilai tunggal dari suatu entitas.



Gambar 2.4 Simbol *Single-value Attribut*

d. *Multi-value Attribute* adalah atribut yang dapat memegang nilai lebih dari suatu entitas.

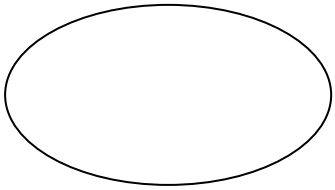


Gambar 2.5 Simbol *Multi-value Attribut*

e. *Derived Attribute* adalah atribut yang mewakili turunan nilai sebuah atribut yang saling berkaitan dan belum tentu dalam tipe entitas yang sama.

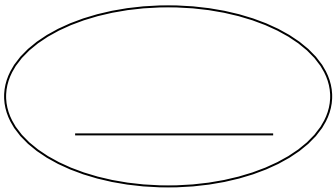
Gambar 2.6 Simbol *Derived Attribut*

f. *Stored Attribute* adalah atribut yang menyimpan nilai yang harus disimpan



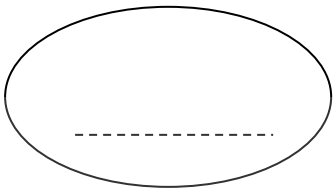
Gambar 2.7 Simbol *Stored Attribut*

g. *Key Attribute* adalah atribut yang menyimpan nilai unik dari setiap *entity* dari strong *entity*.



Gambar 2.8 Simbol *Key Attribut*

h. *Partial Attribute* adalah atribut yang menyimpan sebagian nilai dari *key attribute*, atribut ini dimiliki oleh *weak entity*.



Gambar 2.9 Simbol *Partial Attribut*

3. *Keys*

Beberapa elemen data memiliki nilai yang diperoleh dari entitas tertentu. Elemen penentu dari suatu entitas disebut elemen kunci (key) (Kadir, 2008). Menurut Connolly dan Begg (2002) *keys* terdiri atas beberapa jenis, yaitu:

a. *Candidate Key*

Candidate key merupakan set minimal dari suatu atribut yang secara unik mengidentifikasi setiap *occurrence* dari tipe entitas. *Candidate key* tidak boleh null (kosong).

b. *Primary Key*

Sebuah *candidate key* yang dipilih untuk mengidentifikasi secara unik tiap kejadian pada suatu entitas. *Primary key* harus bernilai *unique* dan tidak boleh *null* (kosong).

c. *Composite Key*

Sebuah *candidate key* yang mempunyai dua atribut atau lebih. Suatu atribut yang membentuk *composite key* bukanlah kunci sederhana karena *composite key* tidak membentuk kunci senyawa.

d. *Alternate Key*

Sebuah *candidate key* yang tidak menjadi *primary key*. *Key* ini biasa disebut dengan *secondary key.*

e. *Foreign Key*

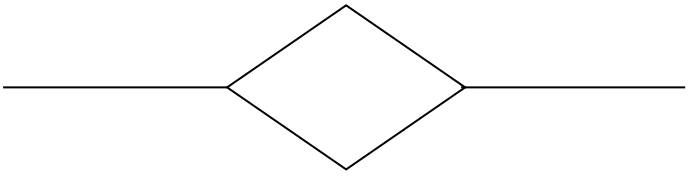
Himpunan atribut dalam suatu relasi yang cocok dengan *candidate key* dari beberapa relasi lainnya. *Foreign key* mengacu pada *primary key* suatu tabel. Nilai *foreign key* harus sesuai dengan nilai *primary key* yang diacunya.

4. *Relationship*

Menurut Whitten (2004) relationship adalah asosiasi bisnis alami antara satu entitas atau lebih. Dalam suatu relasi, entitas yang saling berelasi memiliki kata kerja aktif yang menunjukan bahwa keduanya saling berelasi satu sama lain. Relasi dapat dibedakan menjadi 3 jenis, yaitu:

a. *General Relationship*

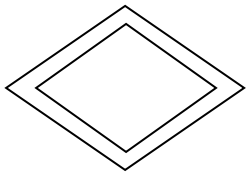
Sebuah relasi yang menghubugkan entitas secara umum, seperti *strong entity* → *strong entity* atau *weak entity* → *weak entity*.



Gambar 2.10 Simbol *General Relationship*

b. *Identyfying Relationship*

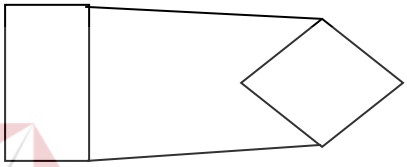
Relasi yang digunakan untuk menghubungkan *stong entity* dengan *weak entity*.



Gambar 2.11 Simbol *Identyfying Relationship*

c. *Recursive Relationship*

Relasi yang digunakan untuk menghubungkan entity yang sama.

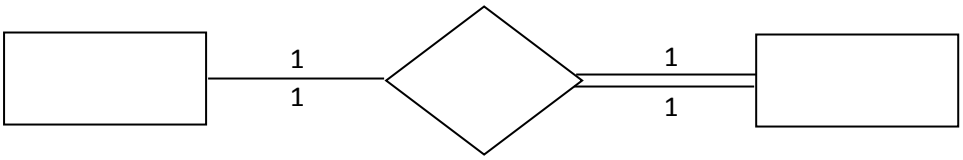


Gambar 2.12 Simbol *Recursive Relationship*

Dalam sebuah relasi antara entity dengan entity yang lain memiliki sebuah *rasio kardinalitas*. *Rasio kardinalitas* memiliki 6 jenis, yaitu:

a. *One to One* (1 : 1)

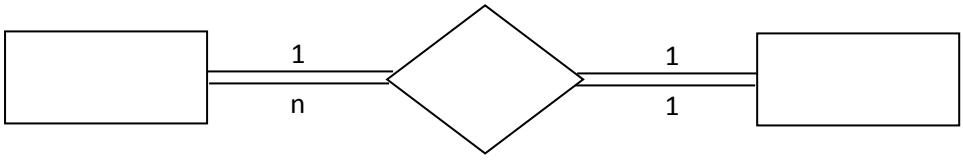
Relasi one to one berarti setiap entitas himpunan A hanya berhubungan dengan satu entitas himpunan B, begitu juga sebaliknya.



Gambar 2.13 Simbol *One to one*

b. *One to Many* (1 : n)

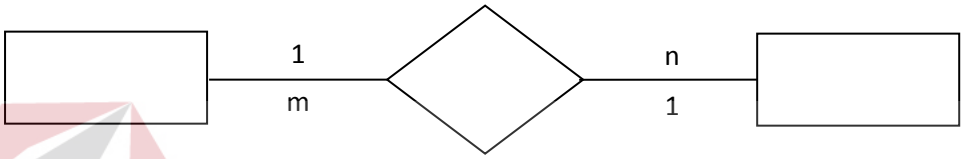
Relasi one to many berarti suatu entitas himpunan A dapat berhubungan dengan banyak entitas pada entitas himpunan B, namun tidak sebaliknya.



Gambar 2.14 Simbol *One to Many*

c. *Many to Many* (m : n)

Relasi *many to many* berarti setiap entitas himpunan A dapat berhubungan dengan entitas pada himpunan B, begitu juga sebaliknya.



Gambar 2.15 Simbol *Many to Many*

d. Relasi rekursif *one to one*

Relasi rekursif one to one adalah sebuah tipe relasi yang dimana entitasnya mempartisipasi lebih dari satu peran.

e. Relasi *superclass/subclass*

Untuk setiap relasi superclass/subclass, entitas superclass diidentifikasikan sebagai entitas induk dan entitas subclass sebagai anggotanya.

f. Relasi *kompleks*

Relasi kompleks adalah tipe relasi yang dimana satu entitas berhubungan dengan entitas lainnya yang dapat membentuk sirkulasi dalam relasi tersebut.

2.12 Siklus Hidup Pengembangan Sistem (SDLC)

Siklus Hidup Pengembangan Sistem atau *Software Development Life Cycle* (SDLC) merupakan suatu bentuk yang digunakan untuk menggambarkan tahapan utama dan langkah–langkah di dalam tahapan tersebut dalam proses

pengembangannya. Kegiatan pengembangan sistem dapat diartikan sebagai kegiatan membangun sistem baru untuk mengganti, memperbaiki atau meningkatkan fungsi sistem yang sudah ada. (Kusrini & Koniyo, 2007)

Menurut Kendall dan Kendall (2010) *system development life cycle* terdiri dari tujuh fase yaitu:

1. Identifikasi masalah, peluang dan tujuan

   Dalam tahap pertama, seorang sistem analis mengidentifikasi masalah, pulang dan tujuan yang akan dicapai. Tahap ini merupakan langkah penting karena menyangkut pengumpulan informasi mengenai kebutuhan konsumen/pengguna.

   Aktivitas dalam tahap ini meliputi observasi atau wawancara kepada narasumber yang berkepentingan, menyimpulkan pengetahuan yang telah diperoleh, mengestimasi cakupan permasalahan/proyek, dan mendokumentasikan hasil-hasil yang telah didapat pada proses sebelumnya.

2. Menentukan kebutuhan informasi

   Pada tahap kedua ini, sistem analis menentukan apa saja kebutuhan informasi yang dibutuhkan oleh pengguna/konsumen. Tahap ini berfokus pada menentukan sample dan memeriksa data mentah, wawancara, observasi pembuat keputusan dan lingkungan perusahaan, serta membuat *prototyping*.

3. Menganalisis kebutuhan sistem

   Tahap ketiga yaitu menganalisis kebutuhan-kebutuhan sistem yang akan digunakan nantinya sebelum proses *coding*. Perangkat yang digunakan dalam tahap ini ialah penggunaan diagram aliran data untuk menyusun *input*, proses dan *output* secara teratur.

4. Merancang sistem yang direkomendasikan

   Dalam tahap ini, sistem analis akan merancang prosedur data *entry* sedemikian rupa sehingga data yang dimasukan kedalam sistem benar-benar sesuai dengan kebutuhan informasi yang telah dibuat sebelumnya. Proses ini berfokus pada rancangan struktur data, arsitektur *software*, representasi *interface*, dan detail (algoritma) prosedural.

5. Mengembangkan dan mendokumentasikan perangkat lunak

   Pada tahap ini merupakan proses membuat kode (*code generation*). *Coding* atau pengkodean merupakan penerjemahan desain dalam bahasa yang bisa dikenali oleh komputer. Programmer akan menerjemahkan transaksi yang diminta oleh user, serta menerjemahkan rancangan sistem yang telah dibuat oleh sistem analis sebelumnya.

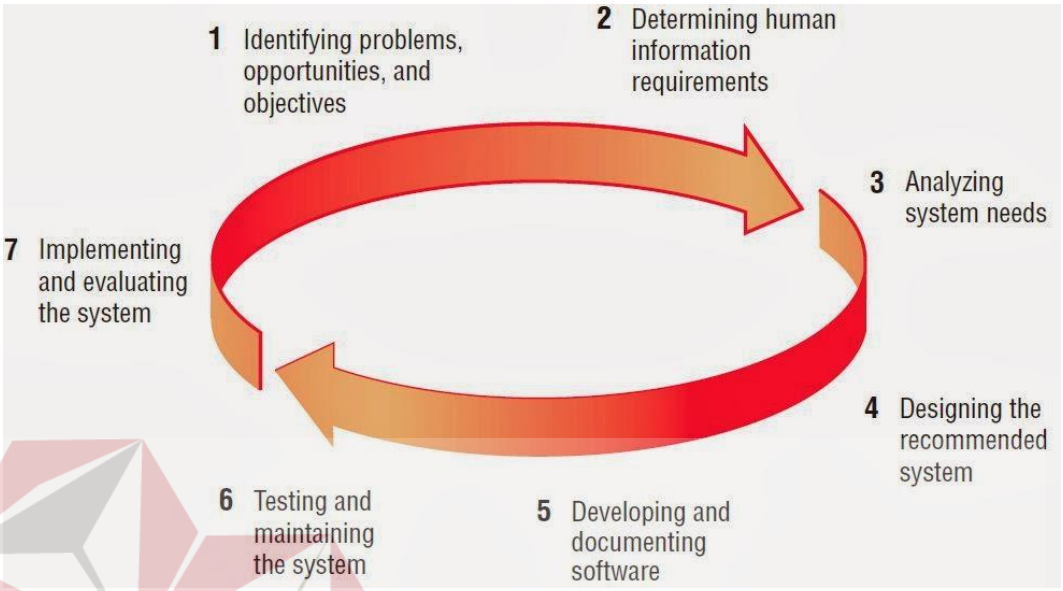
6. Menguji dan mempertahankan sistem

   Sebelum sebuah sistem dapat digunakan, pada tahap ini akan dilakukan pengujian terlebih dahulu. Tujuan dari testing/pengujian adalah menemukan kesalahan-kesalahan terhadap sistem tersebut yang kemudian bisa segera diperbaiki.

7. Mengimplementasikan dan mengevaluasi sistem

   Tahap ini merupakan tahap terakhir dalam pengembangan sistem, yang melibatkan pelatihan bagi pemakai agar dapat menggunakan sistem dengan baik. Selain itu juga, dalam tahap ini sistem analis perlu merencanakan pengembangan sistem untuk kedepannya. Proses ini mencakup pengubahan *file* dari sistem lama ke sistem yang baru.

Dari penjelassan fase-fase SDLC diatas dapat digambarkan sebagai berikut:



Gambar 2.16 *System Development Life Cycle* (Kendall dan Kendall, 2010)