BAB 2

LANDASAN TEORI

2.1 Sistem Informasi

2.1.1 Pengertian Sistem Informasi

Sistem Informasi menurut Laudon (2004, p8) adalah suatu komponen yang saling berhubungan yang bekerja sama untuk mengumpulkan, memproses, menyimpan dan menghilangkan informasi untuk mendukung pengambilan keputusan, koordinasi, pengontrolan, analisis dan visualisasi dalam suatu perusahaan.

Pengertian sistem informasi menurut O'Brien (2006, p6) adalah suatu kombinasi yang terorganisasi yang terdiri dari manusia, perangkat keras, perangkat lunak, jaringan komunikasi, dan data yang mengumpulkan, melakukan transformasi, dan mendistribusikan informasi di dalam suatu organisasi.

Dari pengertian tersebut dapat disimpulkan bahwa pengertian Sistem Informasi adalah sekumpulan komponen pembentuk *system* yang memiliki keterkaitan dan bertujuan untuk menghasilkan informasi dalam bidang tertentu.

2.1.2 Pengertian Sistem Informasi Manajemen

Sistem Informasi Manajemen menurut Laudon dan Laudon (2004, p45) adalah sistem informasi pada tingkat fungsi manajemen dengan menyediakan laporan-laporan untuk manajer atau dengan akses langsung kedalam kegiatan terakhir dan data-data sebelumnya.

Sistem Informasi Manajemen menyediakan informasi dalam bentuk laporan dan menyajkannya bagi manajer dan professional bisnis (O'Brien, 2006, p30).

Berdasarkan pengertian diatas dapat disimpulkan bahwa Sistem Informasi Manajemen adalah sistem informasi yang menyajikan informasi berupa laporan untuk mendukung pihak manajemen.

2.1.3 Peran Fundamental Dari Sistem Informasi Dalam Bisnis

Menurut O'Brien dan Marakas (2006, p6) terdapat tiga alasan fundamental untuk semua aplikasi bisnis dari teknologi informasi. Alasan tersebut dapat ditemukan dalam tiga peran penting yang dapat diberikan sistem informasi pada bisnis proses:

- Mendukung bisnis proses bisnis itu sendiri
- Mendukung pengambilan keputusan dari pegawai dan manajer
- Mendukung strategi perusahaan untuk mencapai keuntungan kompetitif

2.2 Analisis dan Perancangan Sistem

2.2.1 Pengertian Analisis Sistem

Analisis Sistem menurut Mcleod (2001, p190) adalah penelitian terhadap sistem yang telah ada dengan tujuan untuk merancang *system* baru atau diperbarui.

Menurut O'Brien (2006, p350), analisis sistem menggambarkan apa yang sistem perlu lakukan untuk menemukan kebutuhan informasi yang diperlukan oleh pemakai.

Berdasarkan pengertian diatas dapat disimpulkan bahwa Analisis Sistem adalah penelitian yang dilakukan pada sistem yang ada dan menekankan pada masalahnya untuk mendapatkan informasi yang dibutuhkan oleh rancangan sistem baru.

2.2.2 Langkah-langkah Dalam Tahap Analisis

Analisis sistem dapat dibagi menjadi 4 tahap:

1. Analisis pendahuluan
2. Penyusunan usulan pelaksanaan analisis sistem
3. Pelaksanaan analisis sistem
4. Penyusunan laporan hasil analisis sistem

2.2.3 Pengertian Perancangan Sistem

Menurut McLeod (2001, p192) Rancangan Sistem adalah penentuan proses dan data yang diperlukan oleh sistem baru.

Menurut Mulyadi (2001, p51), perancangan sistem adalah suatu proses penerjemahan kebutuhan pemakai informasi ke dalam alternatif rancangan sistem informasi yang diajukan kepada pemakai informasi untuk dipertimbangkan.

Berdasarkan pengertian diatas dapat disimpulkan bahwa perancangan sistem adalah penentuan proses dan data yang diperlukan oleh sistem baru dengan menterjemahkan kebutuhan ke dalam alternatif rancangan untuk dipertimbangkan oleh user.

2.2.4 Langkah-Langkah dalam Tahap Perancangan

Menurut McLeod (2001, p130), langkah-langkah dalam perancangan sistem adalah sebagai berikut :

1. Menyiapkan rancangan sistem yang rinci
2. Mengidentifikasi berbagai alternatif konfigurasi sistem
3. Mengevaluasi berbagai alternatif konfigurasi sistem tersebut
4. Memilih konfigurasi yang terbaik
5. Menyiapkan usulan penerapan

6. Menyetujui atau menolak penerapan sistem

2.3 Persediaan

2.3.1 Pengertian Persediaan

Persediaan dalam suatu perusahaan adalah factor pendukung penting dalam menjalankan operasi. Berikut pendapat para ahli tentang persediaan:

- Persediaan merupakan sejumlah sumber daya yang disimpan untuk memenuhi kebutuhan sekarang maupun kebutuhan yang akan datang. (Render et al, 2006 , p190)
- Menurut Epstein (2004, p208), persediaan merupakan aset pengadaan barang di dalam sebuah bisnis, atau yang sedang dalam proses produksi untuk penjualan tertentu, atau dalam wujud material atau pendukung untuk digunakan dalam proses produksi atau penyumbangan jasa.

Dengan demikian dapat disimpulkan bahwa persediaan merupakan sumberdaya yang dibutuhkan oleh perusahaan, karena pentingnya peranan persediaan barang bagi perusahaan, maka persediaan harusnya dalam pengendalian dan pengawasan yang ketat.

Ada 5 alasan mengapa perlu dilakukannya pengadaan persediaan di dalam perusahaan, yaitu:

- Memungkinkan perusahaan mencapai skala ekonomis
- Mengembangkan adanya persediaan dan permintaan
- Memungkinkan spesialisasi produksi
- Melindungi adanya ketidakpastian permintaan dan siklus pemesanan

- Bertindak sebagai penyangga (buffer) diantara antar muka yang bersifat kritis dalam rantai supply.

Penentuan kebijakan dalam persediaan yang perlu diperhatikan adalah bagaimana perusahaan dapat meminimalkan biaya yang dikeluarkan tersebut berkaitan dengan persediaan barang.

Menurut Nafarin (2002, p186), tujuan pengawasan persediaan, yaitu:

a. Menjaga agar perusahaan tidak sampai kehabisan persediaan sehingga dapat mengakibatkan terhentinya kegiatan produksi
b. Menjaga agar pembentukan persediaan oleh perusahaan tidak terlalu besar atau berlebihan, sehingga biaya persediaan yang timbul tidak terlalu besar
c. Menjaga agar pembelian secara kecil-kecilan dapat dihindari karena akan mengakibatkan timbulnya biaya pemesanan yang besar

2.3.2 *Reorder Point*

Menurut Nafarin (2002, p60), *Reorder Point* adalah saat harus dilakukan pesanan kembali bahan yang diperlukan sehingga kedatangan bahan yang dipesan tersebut tepat pada waktu persediaan di atas *safety stock* sama dengan nol.

Menurut Assauri (2001, p209), *Reorder Point* adalah suatu titik atau batas dari jumlah persediaan yang ada pada suatu saat dimana pesanan harus diadakan kembali.

Menurut Assauri (2001, p199), *lead time* adalah lamanya waktu antara mulai dilakukannya pemesanan bahan-bahan sampai dengan kedatangan bahan-bahan yang dipesan tersebut dan diterima di gudang persediaan.

Menurut Assauri (2001, p198), *safety stock* adalah persediaan tambahan yang diadakan untuk melindungi atau menjaga kemungkinan terjadinya kekurangan bahan (*stock-out*).

Rumus Titik Pemesanan Kembali

Manajer manufaktur dapat menghitung ROP dengan menggunakan rumus berikut:

R = LU + S

Tabel 2.1 Rumus ROP

Keterangan: R = titik pemesanan kembali

L = *Lead Time* pemasok(dalam hari)

U = Tingkat pemakaian (jumlah unit yang diggunakan atau terjual per hari)

S = Tingkat *Safety Stock* (dalam unit)

Misalnya, jika pemasok memerlukan 14 hari untuk menyediakan bahan baku yang dipesan, dan anda menggunakan 10 unit per hari, anda akan menggunakan 140 unit sementara menunggu pemasok memenuhi pesanan. Tambahkan angka ini pada safety stock sebesar 16, dan titik pemesanan kembalinya adalah 156.

2.4 Penjualan

2.4.1 Pengertian Penjualan

Pengertian penjualan menurut Kotler et al (2006, p457), penjualan merupakan sebuah proses dimana kebutuhan pembeli dan kebutuhan penjualan dipenuhi, melalui pertukaran informasi dan kepentingan.

Menurut Syahrul dan Nizar (2000, p716), penjualan adalah pendapatan yang diterima dari pertukaran barang atau jasa dan dicatat untuk satu periode akuntansi

tertentu, baik berdasarkan kas (sebagaimana diterima) atau berdasarkan akrual (sebagaimana diperoleh).

Dengan demikian maka dapat disimpulkan penjualan adalah sejumlah aktivitas yang terjadi antara pembeli dan penjual dengan cara tertentu guna mendapatkan kesepakatan bersama.

Penjualan terdiri dari :

a. Penjualan secara tunai, adalah penjualan yang dilakukan dengan cara mewajibkan pembeli melakukan pembayaran barang terlebih dahulu sebelum barang yang dipesan diserahkan oleh perusahaan kepada konsumen.
b. Penjualan secara kredit, adalah penjualan yang dilakukan dengan menyerahkan barang dipesan, dimana perusahaan hanya menerima sebagian yang dibayarkan dan sisanya diangsur sesuai dengan ketentuan yang telah ditetapkan.

Menurut Yunarto (2006, p6-7), Manajemen penjualan terdiri dari enam proses dasar yaitu aktivitas *pre-sales*, pemrosesan *sales order*, aktivitas *inventory sourcing*, *shipping*, *billing* dan *payment*.

Siklus manajemen penjualan dapat dimulai dari aktivitas pre-sales seperti negosiasi dengan customer yang kemudian disertai dengan pembuatan quotation (semacam penawaran harga). Kemudian prosesnya dilanjutkan dengan pembuatan sales order (order penjualan). Setelah itu perusahaan perlu menyediakan barang yang dipesan oleh customer, perusahaan akan melihat apakah barangnya sudah ada di gudang, perlu diproduksi terlebih dahulu, atau perlu dipesan ke supplier. Hal ini dikenal dengan istilah inventory sourcing. Setelah barang tersedia, lalu dilakukan proses delivery, yaitu guna mengirimkan barang ke customer. Kemudian proses billing akan dilakukan, perusahaan

menagih ke *customer* dengan mengirim *invoice* (faktur). Berdasar *invoice* tersebut *customer* akan melakukan *payment* (pembayaran) atas barang yang dibeli.

2.5 Penerimaan Kas

2.5.1 Pengertian Penerimaan Kas

Menurut Mulyadi (2001, p455), penerimaan kas perusahaan berasal dari dua sumber utama: penerimaan kas dari penjualan tunai dan penerimaan kas dari piutang.

Menurut Mulyadi (2001, p462), informasi yang umumnya diperlukan oleh manajemen dari penerimaan kas adalah :

1. Jumlah pendapatan penjualan menurut jenis produk atau kelompok produk selama
2. jangka waktu tertentu
3. Jumlah kas yang diterima dari penjualan
4. Jumlah harga pokok produk yang dijual selama jangka waktu tertentu
5. Nama dan alamat pembeli
6. Kuantitas produk yang dijual
7. Nama wiraniaga yang melakukan penjualan
8. Otorisasi pejabat yang berwenang

2.6 Pengertian *Object Oriented Analysis and Design (OOAD)*

2.6.1 Pengertian Object Oriented

Pengertian *object* menurut Mcleod (2001, p330) adalah suatu entitas fisik atau kejadian yang dijelaskan dalam bentuk data dan prosesnya. Selain itu *object* juga mempunyai suatu *state* dan suatu sifat *identity*. *Object Oriented* atau orientasi objek merupakan suatu cara untuk melakukan permodelan sistem dengan berorientasikan pada *object-object* yang terlibat dalam sistem tersebut.

Beberapa keuntungan dari *Object Oriented* adalah:

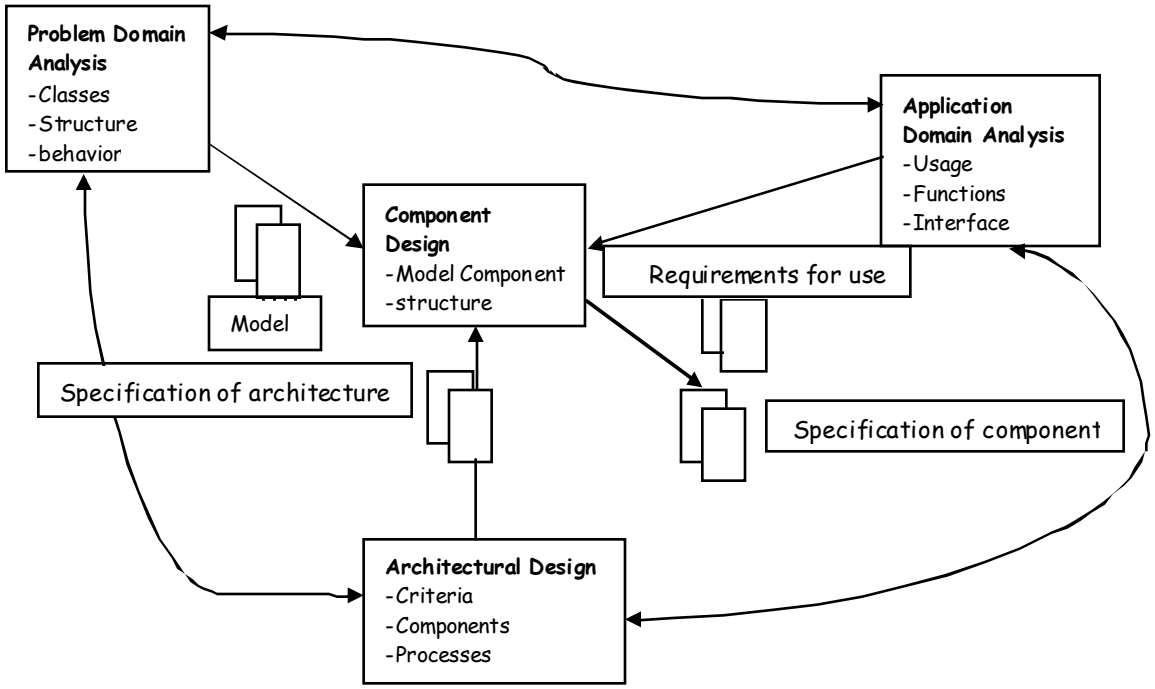
1. Merupakan konsep umum yang dapat digunakan untuk memodelkan hampir semua fenomena yang ada di dunia dengan bahasa umum (*natural language*)
    - *Noun* menjadi *object* atau *class*
    - *Verb* menjadi *behaviour*
2. Dapat digunakan untuk mengembangkan sistem secara incremental
3. Menyediakan informasi yang jelas mengenai *context* dari *system*
4. Mengurangi biaya *maintenance* atau *development*

2.6.2 Definisi OOAD

Object Oriented Analysis menurut Mathiassen et al (2000, p13) adalah aktivitas mengenai persoalan yang diambil secara terpisah dan dijabarkan. *Object Oriented Design* adalah aktivitas yang membangun bagian yang dikenal kemudian disatukan dengan cara yang baru. *Object Oriented Design* mempunyai dua hal penting yaitu:

- *Object Oriented Design* menuntun kepada suatu *object oriented decomposition*
- *Object Oriented Design* menggunakan metode yang berbeda untuk menyatakan perbedaan model-model dari rancangan logika (kelas dan struktur objek) dan fisik (modul dan arsitektur proses) sebuah sistem, di samping aspek statis dan dinamis sebuah sistem.

Perspektif-perspektif tersebut berhubungan dengan empat aktivitas utama dalam OOAD, yaitu : *Problem Domain Analysis, Application Domain Analysis, Arhitectural Design, dan Component Design.*



Gambar 2.1 Aktivitas utama dalam OOA&D
(Mathiassen et.al, 2000, p15)

2.6.3 Keuntungan OOA&D

Dalam analisis dan perancangan tradisional, metode, fungsi, data dan alur data merupakan kunci utama analisis. Namun OOA&D menggunakan objek dan *class*

sebagai kunci utama analisis dan perancangan sistem. Beberapa keuntungan utama lewat penggunaan metode OOA&D adalah:

- OOA&D menyediakan informasi yang jelas mengenai konteks dari sistem. Metode OOA&D memiliki fokus baik pada sistem maupun konteks dari sistem tersebut.
- Metode OOA&D memberikan hubungan yang dekat antara analisis, perancangan, *user interfaces* dan *programming*.

2.6.4 *System Choice*

Menurut mathiassen et al (2000, p25), sebuah proyek pengembangan berawal dari berbagai macam ide yang berbeda tentang sistem yang diinginkan. *System choice* didasarkan pada tiga subaktivitas. Subaktivitas yang pertama dipusatkan pada tantangan-tantangan, kita mencoba mendapatkan kedua gambaran umum dari situasi dan berbagai cara orang-orang menginterprestasikannya. Subaktivitas yang kedua adalah menciptakan dan mengevaluasii ide-ide untuk perancangan sistem. Subaktivitas yang ketiga adalah memformulasikan dan memilih *system definition*, membicarakan dan mengevaluasi alternatif *system definition* dalam hubungannya ke situasi yang kita hadapi.

2.6.5 *System Definition*

Dalam melakukan analisis dan perancangan sistem, langkah awal yang harus dilakukan adalah menggambarkan sistem tersebut terlebih dahulu. *System definition* merupakan deskripsi singkat dari sistem yang terkomputerisasi yang dituliskan dalam bahasa alamiah (Mathiassen et al., 2000, p24). Definisi sistem

menggambarkan konteks sistem, informasi yang harus dimiliki, fungsi-fungsi yang harus disediakan, dimana penggunaan sistem dan kondisi pengembangan yang tepat.

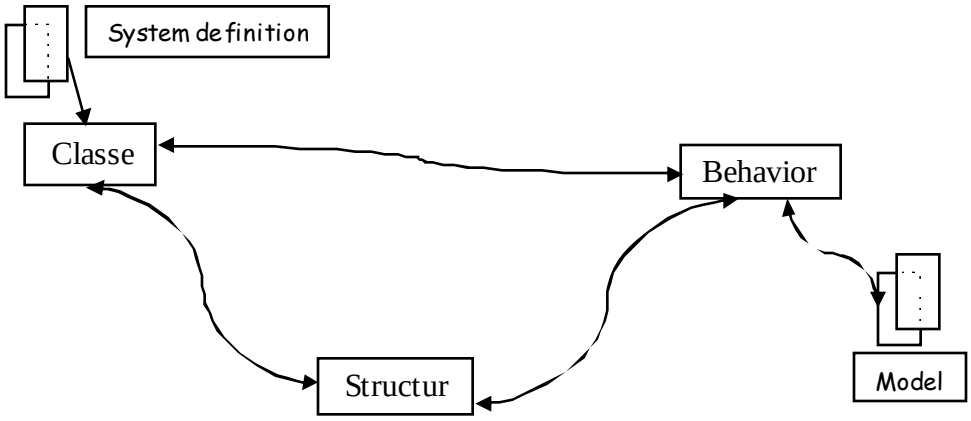
Untuk memudahkan dalam membuat *system definition*, maka dapat digunakan kriteria FACTOR, yaitu :

a. *Functionality* : fungsi-fungsi sistem yang mendukung tugas-tugas dari *application domain*.
b. *Application domain* : bagian dari organisasi yang mengatur, memonitor, dan mengontrol suatu *problem domain*.
c. *Conditions* : kondisi dimana suatu sistem akan dikembangkan dan digunakan.
d. *Technology* : teknologi yang digunakan untuk mengembangkan sistem dan teknologi ketika sistem dijalankan
e. *Objects* : objek-objek utama di dalam *problem domain*.
f. *Responsibility* : tanggung jawab seluruh sistem dalam hubungannya dengan konteks.

2.6.6 *Problem Domain Analysis*

Menurut Matthiassen et al (2000, p6), pada tahap ini dilakukan pengidentifikasian informasi-informasi yang harus ada pada suatu sistem untuk menghasilkan sebuah model sistem. *Problem Domain* merupakan bagian dari keadaaan yang akan diatur, dipantau, dan dikontrol oleh sistem. Sumber dari aktivitas ini adalah *system definition*, yaitu deskripsi singkat dan jelas dari sistem terkomputerisasi dengan menggunakan bahasa alami.

Terdapat tiga subaktivitas dalam *problem domain analysis*, yaitu :



Gambar 2.2 Aktivitas dalam *Problem Domain Analysis*
(Mathiassen et.al, 2000, p46)

1. *Classes*

Menurut Mathiassen et.al. (2000, p4), ini merupakan tahap pemilihan *class* dan *event* dari *system definition* untuk menghasilkan *event table*. *Class* adalah deskripsi dari kumpulan *object* yang mempunyai *structure, behavioural pattern*, dan *attributes* yang sama. Pada tahap analisis, biasanya sebuah *class* cukup didefinisikan dengan namanya saja, tetapi juga dapat ditambah *detail attributes* dan *operation*. *Event* adalah kejadian bersifat instant yang melibatkan *satu object* atau lebih.

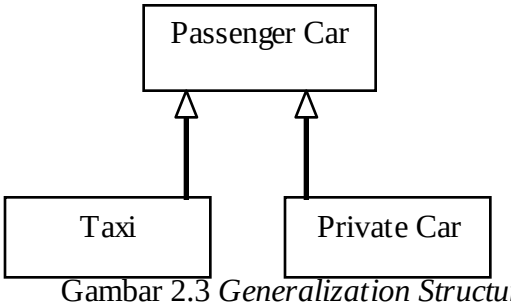
2. *Structure*

Menurut Mathiassen et.al. (2000, p69), tujuan *structure* adalah untuk mendeskripsikan hubungan struktural antara *class* dan *object*. Sumber dari aktivitas ini adalah *event table* dan hasil akhirnya adalah *class diagram* yang menyediakan gambaran ikhtisar dalam *problem domain* yang bertalian secara logis dengan menggambarkan seluruh hubungan struktural antara *classes* dan *objects* di dalam model.

Menurut Mathiassen et.al. (2000, p72) terdapat dua tipe *structure* yaitu :

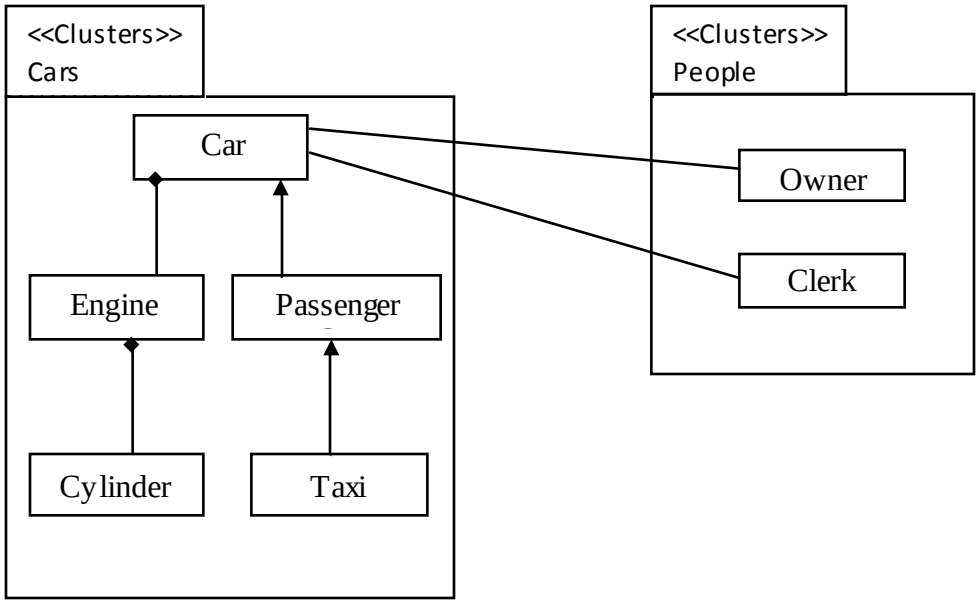
1. *Class Structure*: mengekspresikan hubungan konseptual yang statis antar *class*. *Class Structure* dibagi menjadi dua macam, yaitu:

   a. *Generalization Structure*: *structure* yang menunjukkan hubungan antara dua *subclass* atau lebih dengan satu *superclass* atau lebih. Sebuah *class* yang umum *(superclass)* mendeskripsikan properti umum kepada *group* dari *special class (subclass)* atau terjadi penurunan *attributes* dan *behaviour* dari *superclass*, tetapi *subclass* juga boleh memiliki *attributes* dan *behaviour* tambahan. Dalam ilmu bahasa, *generalization structure* diekspresikan dengan formula "is a".



Gambar 2.3 *Generalization Structure*
(Mathiassen et.al, 2000, p73)

   b. *Cluster Structure*: kumpulan *class–class* yang saling berhubungan. *Cluster* digambarkan dengan notasi *file folder* yang melingkupi *class-class* yang saling berhubungan di dalamnya. *Class-class* dalam satu *cluster* biasanya memiliki hubungan berupa *generalization* atau *aggregation*, sedangkan antara *class* dan *cluster* yang berbeda memiliki hubungan *association*.



Gambar 2.4 *Cluster Structure*
(Mathiassen et.al, 2000, p75)

2. *Object Structure*: mengekspresikan hubungan dinamis dan konkret antar *object*. Hubungan ini dapat berubah secara dinamis tanpa mempengaruhi perubahan pada deskripsinya. Biasanya terdapat *multiplicity* yang menspesifikasikan jumlah dari object yang berelasi. *Multiplicity* dapat berupa *string of numbers* dan penyebaran interval dengan koma seperti "0,3,7,9,..,13,19..*" ; "*" yang disebut *many* ; dan 0..*.

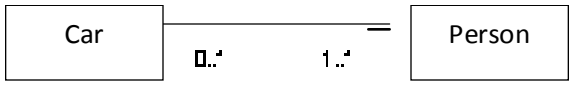
Ada dua tipe *object structure* yaitu:

a. *Aggregation Structure*: mendefinisikan hubungan antara dua *object* atau lebih.

Sebuah *superior object (whole)* memiliki beberapa *object (parts)*. Dalam ilmu bahasa, *aggregation structure* diekspresikan dengan formula *"has a"*, *"a-part-or"*, *"is-owned-by"*. Menurut Mathiassen et.al. (2000, p79) terdapat tiga tipe *aggregation structure* yaitu:

- *Whole part*: dimana *whole* merupakan jumlah dari *parts*, jika salah satu *parts* dihilangkan maka secara tidak langsung telah mengubah *whole*.
- *Container-content*: dimana *whole* adalah *container* (tempat tampung) dari *parts*-nya, jika ada penambahan atau pengurangan terhadap isinya *(parts)* maka tidak akan mengubah pengertian *whole*-nya
- *Union-member*: dimana *whole* merupakan union atau gabungan yang terorganisir dari anggotanya *(parts)*, jika ada penambahan atau pengurangan terhadap anggota maka tidak akan mengubah *union*-nya. Ada batasan jumlah anggota terendah karena tidak mungkin sebuah *union* tanpa anggota.

b. *Association Structure*: mendefinisikan hubungan antara dua *object* atau lebih tetapi berbeda dengan *aggregation* (Mathiassen et.al, 2000, p76). Hubungan antar *class* pada *aggregation* memiliki pertalian yang kuat, sedangkan pada *association* tidak. Dalam ilmu bahasa, *association structure* diekspresikan dengan formula "*knows*" atau "*associated-with*".



Gambar 2.5 *Association Structure*
(Mathiassen et.al, 2000, p77)

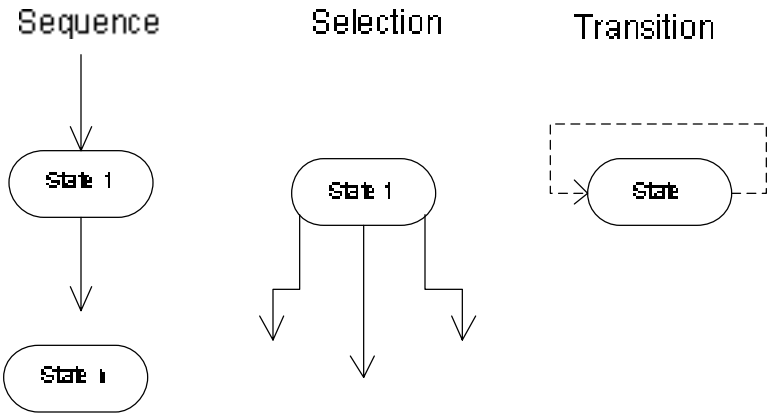
3. *Behavior*

Menurut Mathiassen et.al. (2000, p89), aktivitas ini bertujuan untuk memodelkan keadaan *problem domain* yang dinamis dengan memperluas definisi *class* yang ada dalam *class diagram* dengan menambahkan *behavioural pattern* dan *attributes* untuk

setiap *class*. Sumber dari tahap ini adalah *event table* dan *class diagram* yang telah dihasilkan dari tahap-tahap sebelumnya, dimana hasil akhirnya adalah *behavioural pattern* yang diekspresikan secara grafis dalam *statechart diagram* (Mathiassen et.al, 2000, pp80-90). *Behaviour* dari *object* dapat didefinisikan dengan *event trace*. *Event trace* adalah serangkaian *event* yang berurutan meliputi suatu *object*. Setiap *object* memiliki *event trace* yang mungkin berbeda-beda walaupun *object-object* tersebut berada dalam *class* yang sama. *Behavioral pattern* adalah deskripsi dari semua *event trace* yang mungkin untuk semua *object* dalam *class*. Untuk menspesifikasikan data-data yang perlu disimpan oleh sistem dalam *problem domain* diperlukan *attributes* yang merupakan deskripsi property dari *class* atau *events* (Mathiassen et.al, 2000, p92).

Menurut Mathiassen et.al. (2000, p93), *behavioral pattern* memiliki struktur kontrol sebagai berikut:

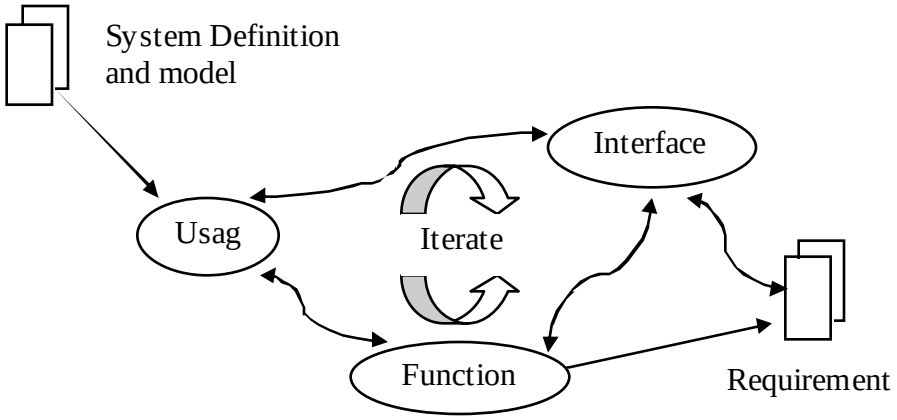
- *Sequence* : suatu set *events* yang terjadi satu per satu. Notasinya: "+".
- *Selection* : satu set *event* yang terjadi akibat suatu set *events* dengan kondisi tertentu. Notasinya: "|".
- *Iteration* : satu *event* yang terjadi berulang-ulang. Notasinya: "*". Setelah *event trace* sudah ditentukan, maka dapat dibuat *behavioral pattern* dalam bentuk *statechart diagram*.



Gambar 2.6 Struktur Kontrol *Statechart Diagram*
(Mathiassen et.al, 2000, p95)

2.6.7 *Application Doman Analysis*

Menurut Matthiassen et al (2000, p6), tahap ini mendefinisikan *requirement* dari suatu sistem. *Application Domain* merupakan bagian yang mengatur, memantau atau mengontrol *Problem Domain* atau dengan kata lain, berhubungan dengan aktivitas yang dikerjakan/dijalankan oleh sistem. Prinsip dari *Application Doman Analysis* adalah bekerja sama dengan *user* untuk menggunakan *usage*, *function* dan *interface*. Sumber dari aktivitas ini adalah *system definition* dan model dari tahap sebelumnya.



Gambar 2.7 Aktivitas dalam *Application Domain Analysis*
(Mathiassen et.al, 2000, p95)

Menurut Mathiassen et.al. (2000, p117), terdapat tiga subaktivitas dalam *application domain analysis*, yaitu :

1. *Usage*

Menurut Mathiassen et.al. (2000, p119), hasil akhir dari aktivitas ini adalah membuat deskripsi dari *actors* dan *use cases*, dimana relasinya diekspresikan dengan menggunakan *actor table* atau *use case diagram*. *Actor* adalah abstraksi *user* atau sistem lain yang berinteraksi dengan sistem. *Use case* adalah pola interaksi antara sistem dengan *actors* dalam application domain.

2. *Functions*

Menurut Mathiassen et.al. (2000, p138), tujuan dari aktivitas ini adalah untuk menentukan kemampuan pemrosesan dari suatu sistem sehingga menghasilkan suatu *function list* dengan spesifikasi untuk *function* yang kompleks. *Function* memfokuskan pada apa yang bisa dilakukan oleh sistem untuk membantu *actor* dan sebagai fasilitas agar model berguna bagi *actor*. Sumber untuk mengidentifikasikan *function* berasal dari deskripsi *problem domain* yang diekspresikan oleh *class* dan *events*, juga berasal dari deskripsi *application domain* yang diekspresikan oleh *use case*. Ada empat tipe *function*, yaitu :

a. *Update* : *function* yang diaktifkan oleh *event* dari *problem domain* dan berakibat pada perubahan status model.

b. *Signal* : *function* yang diaktifkan oleh perubahan dalam status model dan bereaksi dalam konteks. Reaksi ini dapat berupa tampilan kepada *actor* dalam *application domain*, atau intervensi langsung ke *problem domain*.

c. *Read* : *function* yang diaktifkan oleh kebutuhan akan informasi dalam tugas *actor* dan berakibat dalam tampilan bagian tertentu dalam model.

d. *Compute* : *function* yang diaktifkan oleh kebutuhan informasi dalam tugas *actor* dan terdiri dari penghitungan yang melibatkan informasi yang disediakan oleh *actor* atau model. Hasilnya adalah perhitungan. Semua *function* yang ada dalam *application domain* akan dimasukkan ke dalam *function list.*

3. *Interfaces*

Menurut Mathiassen et.al. (2000, p151), tujuan dari aktivitas ini adalah untuk menentukan antar muka *(interface)* sistem dari sistem yang sedang dikembangkan. *Interface* adalah fasilitas yang membuat model sistem dan *function* tersedia bagi *actor*. Dengan adanya *interface* memungkinkan *actor* untuk berinteraksi dengan sistem. Sumber aktivitas ini berasal dari *class diagram*, *use cases*, dan *function list.*

Menurut Mathiassen et.al. (2000, p152), terdapat dua tipe *interface* yaitu:

a. *User Interface*: menghubungkan *human actor* (manusia) dengan sistem.

Dalam merancang *user interface* dibutuhkan *feedback* dari *user*. Ada empat tipe user *interface pattern*, yaitu:

- *Menu-selection* : *interface* yang memberikan daftar pilihan
- *Form fill-in* : *interface* yang digunakan untuk *entry data*
- *Command language* : *interface* yang digunakan dengan memasukkan perintah dan memerlukan daya ingat *user* untuk mengoperasikan sistem
- *Direct manipulation pattern:* yang membiarkan *users* bekerja dengan menggunakan representasi-representasi *object*

b. *System Interface* : menghubungkan *system actor* (sistem lain) dengan sistem yang sedang dikembangkan. Sistem lain bisa berupa *external device* (sensor, switch, *printer*, dll) dan sistem komputer yang kompleks sehingga dibutuhkan suatu protokol komunikasi.

Menurut Mathiassen et al (2000, p156-158), untuk menentukan elemen dari user interface dapat menggunakan object dan class pada model serta functions. Elemen tersebut harus direpresentasikan dalam bentuk yang mudah dipahami oleh user, seperti *icon, fields, tables, diagrams, windows,* dan *button*. Untuk analisa yang lebih kompleks dalam menentukan elemen dari *user interface* dapat menggunakan *sequence diagram* untuk merelasikan interaksi antara elemen *interface* dengan *use case*-nya. *Sequence Diagram* mendeskripsikan langkah-langkah interaksi *individual* dan menghubungkannya dengan window yang relevan. Diagram ini juga menggambarkan *functions* yang akan diaktivasi selama interaksi terjadi.

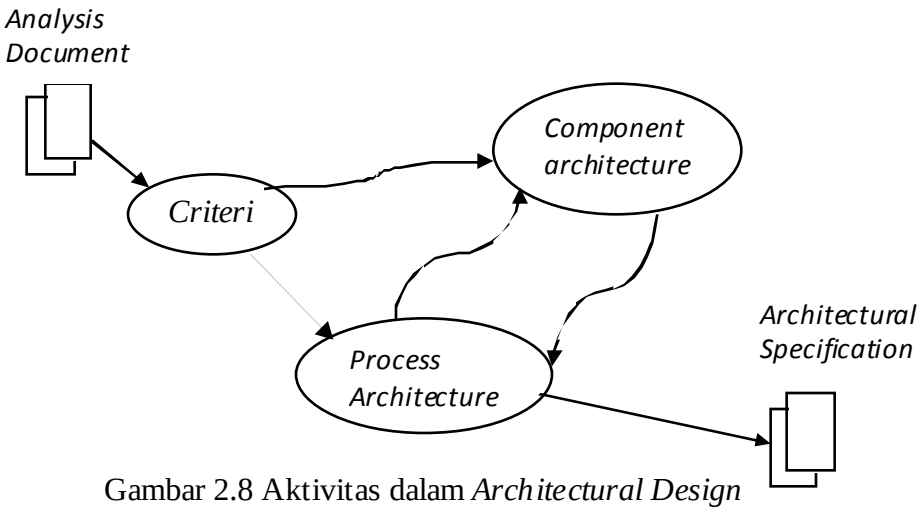
Menurut Mathiassen et al (2000, p156-158), deskripsi dari *user interface* dapat menggunakan *navigation diagram* yang menyediakan gambaran menyeluruh dari elemen *user interface* dan transisi di antaranya. Diagram ini terdiri dari gambar yang diperkecil di setiap window, tanda panah yang menunjukkan bagaimana menggunakan *button*, dan seleksi lain yang akan mengaktivasi *function* atau membuka window lain.

Menurut Mathiassen et al (2000, p159-161), untuk menggambarkan elemen-elemen *user interface* dalam *prototype* atau menspesifikasikannya lebih detail dapat menggunakan window diagram. Diagram ini mendeskripsikan tampilan dari single window yang mencakup bentuk detail dari elemen-elemen window.

2.6.8 *Architectural Design*

Menurut Mathiassen et al (2000, p173), pada aktivitas ini akan dilakukan penstrukturan sistem berdasarkan bagian-bagiannya dan pemenuhan beberapa *criteria* desain. Aktivitas ini juga merupakan suatu *framework* bagi aktivitas pengembangan selanjutnya. Aktivitas ini bertujuan untuk menstrukturkan suatu sistem yang terkomputerisasi. Hasil yang diperoleh berupa struktur dari komponen-komponen dan proses-proses sistem.

Tahap ini memiliki tiga subaktivitas yaitu :



Gambar 2.8 Aktivitas dalam *Architectural Design*
(Mathiassen et.al, 2000, p176)

1. *Criteria*

Menurut Mathiassen et al (2000, p176), criteria adalah suatu prioritas dari arsitektur. Tujuan aktivitas ini adalah untuk menentukan prioritas desain. Hasil yang diperoleh dari tahap ini adalah kumpulan criteria untuk desain yang telah diprioritaskan.

CRITERIA	PENGUKURAN DARI
Usable	Kemampuan adaptasi sistem terhadap konteks organisasi, hubungan kerja, dan teknikal

Secure	Pencegahan melawan akses yang tidak terotorisasi terhadap fasilitas-fasilitas yang ada
Efficient	Penggunaan yang ekonomis terhadap fasilitas technical platform
Correct	Pemenuhan terhadap persyaratan - persyaratan
Reliable	Pemenuhan terhadap eksekusi function yang benar-benar tepat
Maintainable	Besarnya usaha untuk melokasikan dan memperbaiki kecacatan sistem
Testable	Besarnya usaha untuk memastikan bahwa sistem menampilkan fungsi – fungsi yang telah ditentukan
Flexible	Besarnya usaha untuk memodifikasi sistem
Comprehensible	Usaha yang diperlukan untuk mendapatkan pengertian yang masuk akal terhadap sistem
Reuseable	Potensi penggunaan bagian – bagian dalam sistem lain yang terhubung
Portable	Besarnya usaha untuk memindahkan sistem ke technical platform
Interoperable	Besarnya usaha untuk menggabungkan suatu sistem ke sistem lain

Tabel 2.2 Criteria klasik untuk mengukur kualitas *software*
(Mathiassen et.al, 2000, p178)

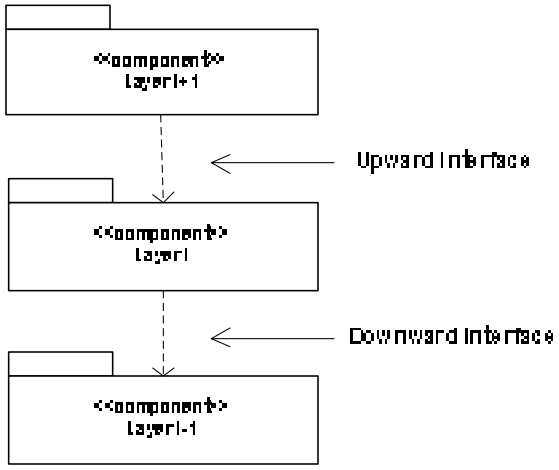
2. *Component*

Menurut Mathiassen et al (2000, p190), *component architecture* adalah sebuah struktur sistem yang terdiri dari komponen-komponen yang saling terhubung. *Component* adalah kumpulan dari bagian-bagian yang membentuk sistem dan memiliki tanggung jawab yang telah terdefinisikan dengan jelas.

Menurut Mathiassen et al (2000, pp193-198), terdapat tiga tipe pola arsitekur umum yang dapat digunakan ketika mendesign suatu *component architecture*, yaitu :

- *Layered Architecture Pattern*

  Arsitektur ini terdiri dari beberapa *components* yang didesain sebagai *layers*. Design dari setiap *component* menggambarkan tanggung jawabnya masing-masing serta *interface* bagian atas maupun bagian bawah. *Interface* bagian atas akan menggambarkan operasi yang tersedia untuk *layer* di bawahnya.

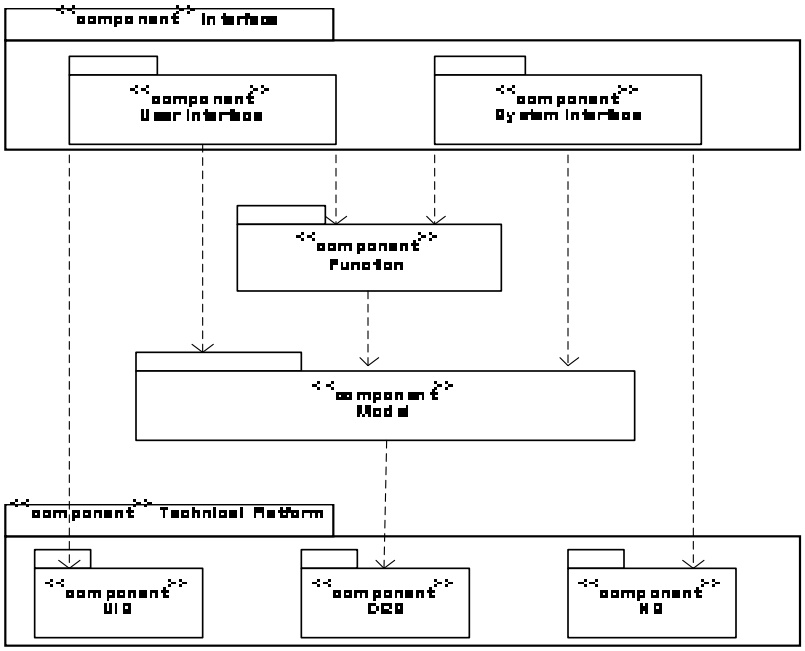


Gambar 2.9 *Layered Architecture Pattern*
(Mathiassen et.al, 2000, p195)

- *Generic Architecture Pattern*

  Model component mengandung model dari sistem *object*, yang dapat berupa *layer* yang paling bawah, kemudian diikuti dengan layer sistem function, dan

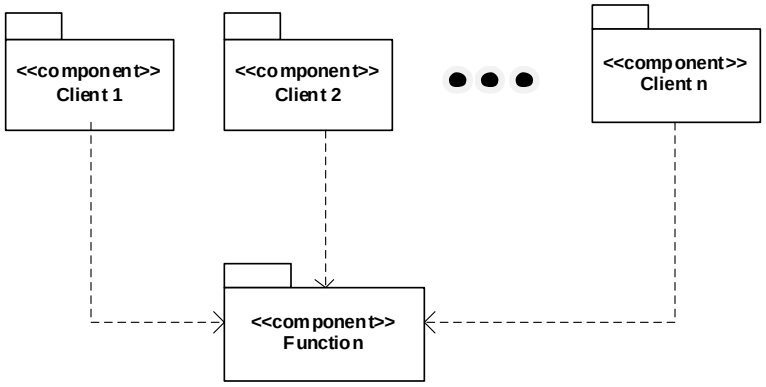
yang paling atas adalah *component interface*. *Layer interface* dapat dibagi menjadi dua bagian, yaitu: *user interface* dan *system interface*.



Gambar 2.10 *Generic Architecture Pattern*
(Mathiassen et.al, 2000, p196)

- *Client-Server Architecture Pattern*

  Komponen arsitektur sebuah *server* dan beberapa *clients*. *Server* memiliki kumpulan operasi yang tersedia bagi *client*. *Server* bertanggung jawab untuk menyediakan hal-hal yang umum bagi *client*-nya, seperti: *database* atau sumber daya lain yang bisa digunakan bersama. *Server* menyediakan operasi untuk *client* melalui suatu jaringan. *Client* bertanggung jawab untuk menyediakan interface lokal bagi para user.



Gambar 2.11 *Client - Server Architecture Pattern*₂
(Mathiassen et.al, 2000, p197)

3. *Process*

Menurut Mathiassen et.al. (2000, p209, pp211-212), tahap ini menentukan bagaimana suatu proses sistem didistribusi dan dikoordinasi. Tujuan dari tahap ini adalah mendefinisikan struktur fisikal dari suatu sistem. Hasil akhirnya adalah *deployment diagram* yang menunjukkan *processor* dengan komponen program dan *active object* yang ditugaskan. *Processor* adalah suatu bagian peralatan yang dapat mengeksekusi sebuah program.

Menurut Mathiassen et.al. (2000, pp216-219), terdapat tiga pola umum deployment diagram, yaitu :

a. *Centralized Pattern*

Pola ini menyimpan semua data di *server* pusat dan para *client* hanya memilki *user interface*. Semua *request* dan *update* dilakukan ketika ada panggilan dari *client* ke *server*.

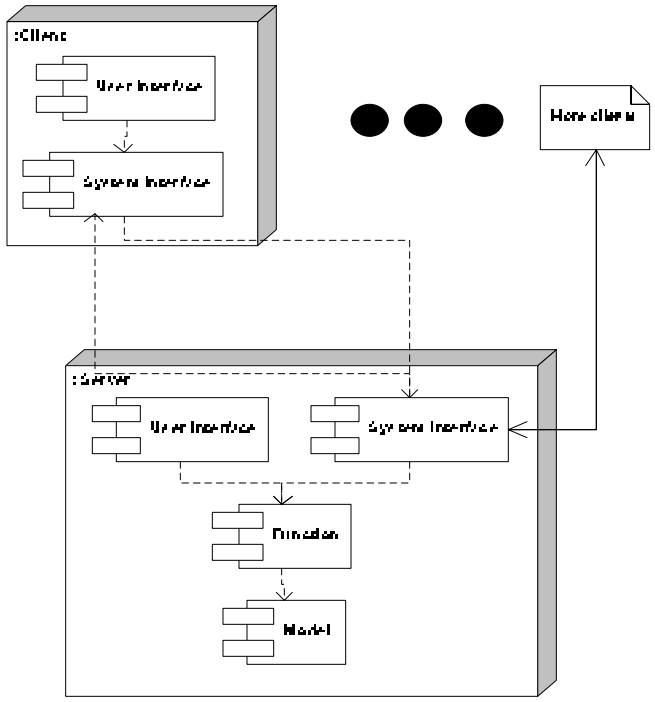
Kelebihan dari pola ini yaitu :

- Tingkat persyaratan teknis client yang tidak begitu tinggi
- Konsistensi data yang terjamin

- Mudah dimengerti dan diimplementasikan
- Lalu lintas jaringan yang sedang

Kelemahan dari pola ini yaitu :

- Kesiapan yang kurang baik
- Waktu akses yang lebih lama
- Pola ini cocok digunakan pada sistem yang lebih tersentralisasi.



Gambar 2.12 *Centralized Pattern*
(Mathiassen et.al, 2000, p216)

b. *Distributed Pattern*

Pola ini mendistribusikan data ke seluruh *client* dan *server* hanya dibutuhkan untuk menyiarkan model *updates* antar *client*

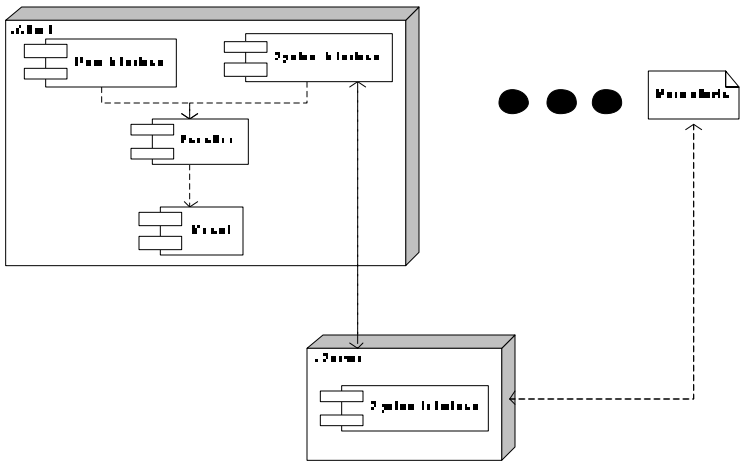
Kelebihan dari pola ini yaitu :

- Waktu akses yang lebih cepat, karena *function* dan *model* ada pada *local client*
- Kesiapan yang lebih matang, karena *client* dapat bekerja tanpa jaringan

Kelemahan dari pola ini yaitu :

- Data yang berulang, karena ada di beberapa *client*
- Potensi inkonsistensi data di antara data dalam *client*
- Persyaratan teknis yang lebih tinggi pada *client*, karena harus menjalankan *model, function,* dan *interface*

Pola ini cocok digunakan apabila sistem memliki model mudah dan function yang rumit.



Gambar 2.13 *Distributed Pattern*
(Mathiassen et.al, 2000, p217)

c. *Decentralized Pattern*

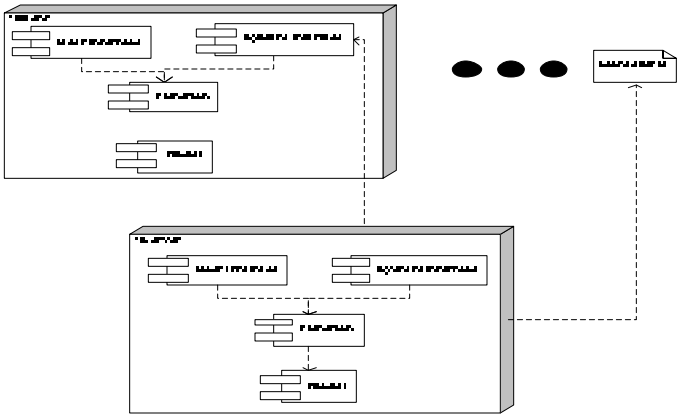
Pola ini memberi konsep dimana client memilki datanya sendiri, sehingga data umum ada di server.

Kelebihan dari pola ini yaitu :

- Konsistensi data yang lebih baik, karena tidak ada duplikasi data antar *client* atau antara *client* dan *server*
- Beban jaringan yang lebih ringan, karena jaringan hanya digunakan ketika data umum di *server diupdate.*
- Waktu akses data lokal lebih cepat, meskipun akses ke data umum lebih lama

Kelemahan dari pola ini adalah semua *processor* harus mampu menjalankan function yang kompleks dan memaintain modal yang besar.

Pola ini lebih cocok digunakan jika *problem domain* dapat dibagi menjadi beberapa *subdomain* yang sama, dimana setiap *subdomain* akan menjadi *application domain* bagi *client.*

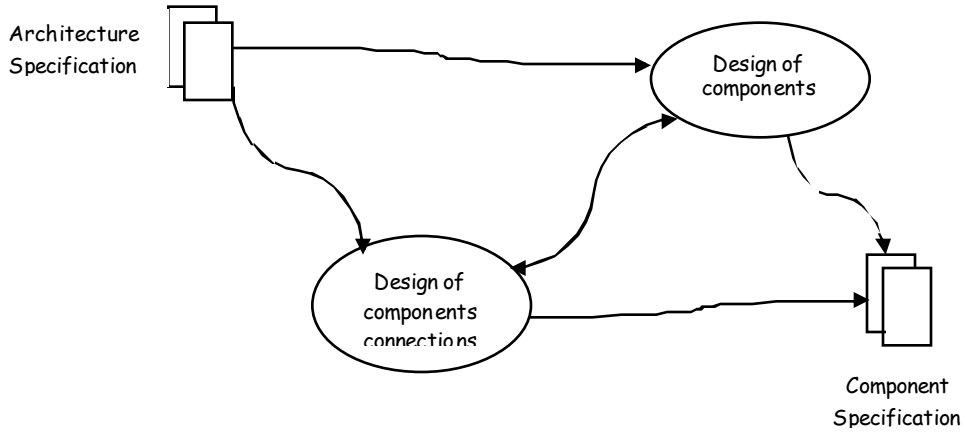


Gambar 2.14 *Decentralized Pattern*
(Mathiassen et.al, 2000, p219)

2.6.9 *Component Design*

Menurut Mathiassen et.al.(2000, p232), tujuan aktivitas ini adalah untuk menentukan implementasi dari kebutuhan di dalam kerangka arsitektur. Sumber dari tahap ini adalah *architectural specification* dan *system requirement* yang akan

menghasilkan *connected component specification*. Ada dua subaktivitas dalam *component design*, yaitu:



Gambar 2.15 Subaktivitas dalam *Component Design*
(Mathiassen et.al, 2000, p232)

1. *Design of Components*

Merupakan tahapan untuk merancang komponen sistem, yaitu:

a. *Model Component*

Menurut Mathiassen et.al. (2000, p236), *model component* adalah bagian dari sistem yang mengimplementasi model *problem domain*. Tujuan dari *model component design* adalah untuk menggambarkan *model* dari *problem domain*. Hasil akhir dari aktivitas ini adalah *class diagram* yang telah direvisi dari hasil kegiatan analisis. Revisi *class diagram* dapat dilakukan dengan memperhatikan *private events* dan *common events*. *Private events* adalah *event* yang hanya melibatkan satu *object domain*.

Event-event yang hanya terjadi secara berurutan atau	Representasikan *event-event* ini sebagai *state attribute* pada *class* yang dijabarkan oleh *statechart diagram*. Setiap kali ada kejadian yang melibatkan

sequence dan *selection*	salah satu *event* tersebut, maka sistem akan menugaskan yang baru kepada *state attribute*.
	Integrasikan *attribute* dari *event* yang terlibat ke dalam *class*
Event-event yang terjadi secara berulang-ulang (*iteration*)	Representasikan *event-event* ini sebagai suatu *class* baru dan hubungkan class tersebut dengan *class* yang dijabarkan pada *statechart diagram* dengan menggunakan struktur *aggregation*. Untuk setiap iterasi, sistem akan menghasilkan suatu *object* baru.
	Integrasikan *attribute event* ke dalam *class* yang baru.

Tabel 2.3 Panduan dalam merepresentasikan *private events*
(Mathiassen et.al, 2000, p241)

Jika suatu *event* adalah *common* atau umum sehingga mempengaruhi beberapa *object*, maka *event* tersebut perlu dihubungkan dengan salah satu *object* dan dibuat hubungan struktural dengan *object* lain agar tetap dapat mengaksesnya.

Common event	Jika *event* yang terlibat dalam *statechart diagram* dalam cara yang berbeda, representasikan *event* tersebut dalam hubungan ke *class* yang menawarkan representasi paling simpel
	Jika *event* yang terlibat dalam *statechart diagram* dalam cara yang sama, pertimbangkan alternatif representasi yang mungkin dapat digunakan

Tabel 2.4 Panduan dalam merepresentasikan *common events*
(Mathiassen et.al, 2000, p241)

b. Function Component

Menurut Mathiassen et.al. (2000, p252), *function component* adalah bagian sistem yang mengimplementasikan kebutuhan fungsional. Tujuannya adalah agar *user interface* dan komponen-komponen sistem lainnya dapat mengakses *model*. Sedangkan tujuan dari *function component design* adalah menentukan implementasi *functions*. Hasil dari kegiatan ini adalah *class diagram* dengan dan spesifikasi dari *operations* yang kompleks. Langkah pertama yang harus dilakukan adalah mendesain *functions* sebagai *operations*, yaitu mengidentifikasi tipe utama dari *functions* tersebut. Menurut Mathiassen et.al. (2000, pp255-260), terdapat empat tipe functions yaitu: *read, update, compute*, dan *signal*. Menurut Mathiassen et.al. (2000, p260), *patterns* (pola) dapat membantu memilih *functional design* mana yang dapat digunakan dari beberapa pilihan yang bisa membantu merealisasikan *functions* sebagai sekumpulan *operations*.

Ada empat pola yang dapat digunakan yaitu:

- *Model Class Placement*

Pola ini menempatkan *operation* dalam *model component class* dan berguna ketika sebuah *operation* hanya mengakses sebuah *single object* atau struktur *aggregation* yang sederhana. Pola ini juga dapat digunakan ketika beberapa *object* terlibat namun hanya jika tanggung jawab *operation* tersebut dapat dengan jelas ditempatkan pada salah satu dari *model class*.

- *Function Class Placement*

Pola ini digunakan ketika tanggung jawab operation tidak dapat dengan jelas ditempatkan pada *model class*. Sebaliknya satu atau lebih *functional-component*

class dapat digambarkan dengan menempatkan *operation* yang merealisasikan *function.*

- *Strategy*

Pola ini digunakan untuk mendefinisikan sekumpulan *operations* yang umum terenkapsulasi dan dapat dipertukarkan.

- *Active Function*

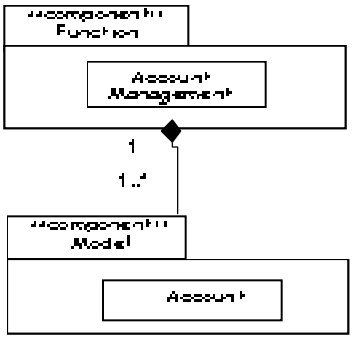
Active signal function dapat direalisasikan sebagai *operation* yang secara permanen aktif dan berkala memberikan sinyal kepada *interface. Active function* ditempatkan sebagai *active-object* dan *perfomance*-nya tergantung dari *state* pada *model component.*

2. *Connecting components*

Tujuan dari aktivitas ini adalah menghubungkan komponen-komponen sistem yang akan menghasilkan *class diagram* dari komponen-komponen tersebut. Jadi pada aktivitas ini, hubungan antara komponen-komponen dirancang untuk mendapatkan desain yang fleksibel dan *comprehensible.* Untuk itu maka dibutuhkan evaluasi dari *coupling* dan *cohesion*. *Coupling* adalah ukuran tentang seberapa dekat dua *classes* atau *components* dihubungkan. *Cohesion* adalah ukuran tentang seberapa baik sebuah *class* atau *component* terikat bersama. Prinsipnya adalah: "*highly cohesive classes and loosely coupled components*". Hasil akhir dari aktivitas ini adalah *class diagram* yang *dependencies*-nya berubah menjadi *connections*. Menurut Mathiassen et.al (2000, p275), terdapat tiga bentuk *connections*, yaitu:

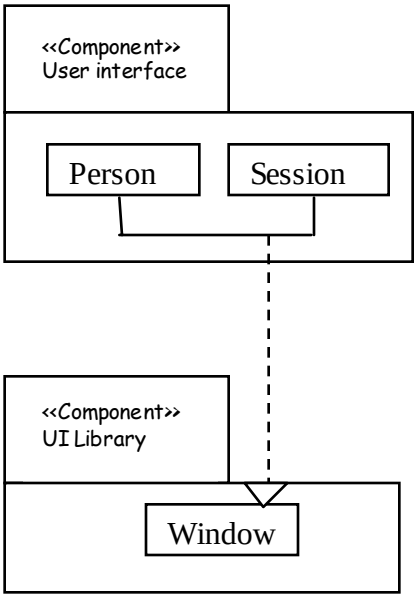
- *Class aggregation*: mengagregasikan *class-class* dari *component* lain. Koneksi ini berguna ketika class definition sudah ada di dalam *component*

lain. Pada umumnya *coupling*-nya rendah, tetapi sulit mencapai *cohesive*. Contoh:



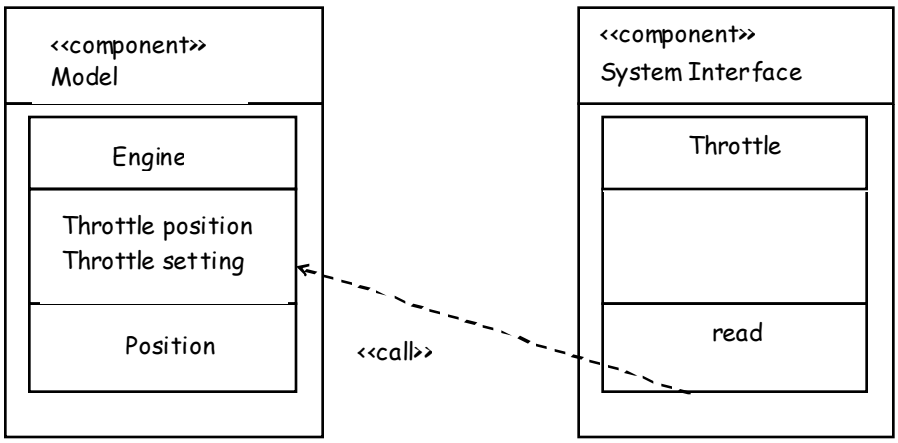
Gambar 2.16 Koneksi oleh *class aggregation*
(Mathiassen et.al, 2000, p275)

- *Class specialization*: menspesialisasikan *public class* dari *component* lain. Contoh:



Gambar 2.17 Koneksi oleh *class specialization*
(Mathiassen et.al, 2000, p276)

- *Operation call*: memanggil *public operations* di dalam *object-object* dari *component* lain. Pada umumnya *coupling*-nya rendah tetapi *cohesion*-nya tinggi. Contoh:



Gambar 2.18 Koneksi dalam memanggil sebuah operasi

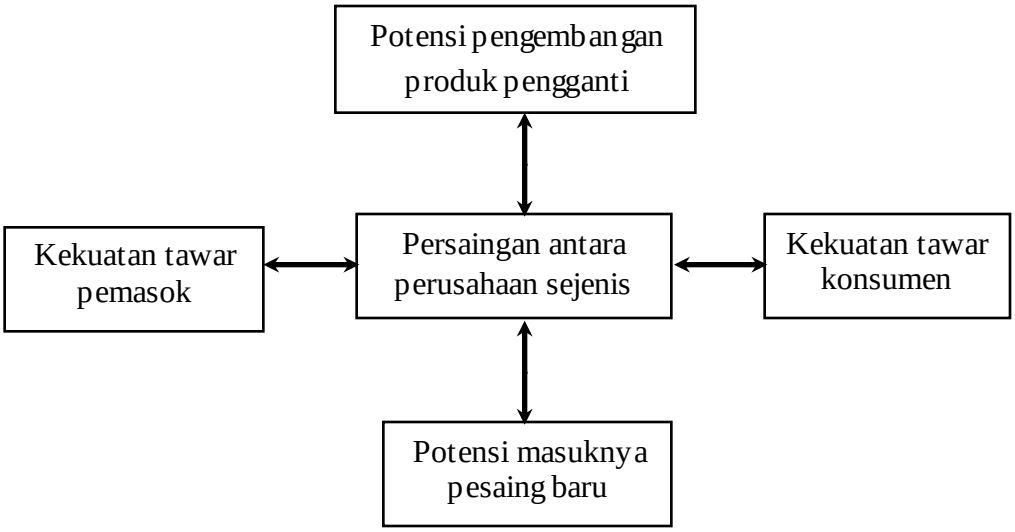
(Mathiassen et.al, 2000, p277)

2.7 Teknik Analisis Data

2.7.1 Model Lima Kekuatan Porter

Menurut David (2004, p144) Analisis Persaingan Model Lima Kekuatan Porter merupakan pendekatan yang banyak dipakai untuk mengembangkan strategi oleh banyak industry. Intensitas persaingan antar perusahaan sangat beragam di berbagai industri. Menurut Porter, sifat persaingan dalam suatu industry dapat dilihat sebagai gabungan dari lima kekuatan berikut ini:

1. Persaingan antara perusahaan sejenis
2. Potensi masuknya pesaing baru
3. Potensi pengembangan produk pengganti
4. Kekuatan tawar pemasok
5. Kekuatan tawar konsumen



Gambar 2.19 Model Lima Kekuatan Persaingan
(David, 2004, p145)

- **Persaingan Antara Perusahaan Sejenis**

Kekuatan ini paling berpengaruh dibandingkan dengan empat kekuatan lainnya. Strategi yang dijalankan oleh satu perusahaan dapat berhasil jika strategi itu memiliki keunggulan kompetitif (*competitive advantage*) dibandingkan dengan strategi yang dijalankan oleh perusahaan pesaing. Perubahan strategi di sebuah perusahaan dapat diimbangi serangan balasan, seperti menurunkan harga, meningkatkan mutu, menambah fitur, menyediakan pelayanan, memperpanjang garansi, dan meningkatkan iklan.

Intensitas persaingan di antara perusahaan yang bersaing cenderung meningkat ketika jumlah pesaing bertambah, ketika perusahaan yang bersaing menjadi setara besarnya dan kemampuannya, ketika permintaan produk industri menurun, dan ketika potongan harga menjadi biasa. Persaingan juga bertambah jika konsumen dapat dengan mudah berganti merek; jika hambatan untuk meninggalkan pasar tinggi; jika produk mudah

rusak; jika perusahaan pesaing memiliki strategi, asal dan budaya yang berbeda; serta jika merger dan akuisisi biasa terjadi dalam industri. Ketika persaingan di antara perusahaan meningkat, laba industri menurun, dan dalam beberapa kasus sampai industry tersebut menjadi tidak menarik.

- **Potensi Masuknya Pesaing Baru**

Ketika Perusahaan baru dapat dengan mudah masuk ke industri tertentu, sudah pasti intensitas persaingan di antara perusahaan meningkat. Hambatan-hambatan terhadap masuknya pesaing baru bisa berupa pentingnya memperoleh skala ekonomi dengan cepat, pentingnya memperoleh teknologi dan pengetahuan khusus, kurangnya pengalaman, kuatnya loyalitas pelanggan, fanatisme terhadap merek tertentu, persyaratan model yang besar, kurangnya saluran ditribusi yang memadai, kebijakan peraturan pemerintah, tarif, kurangnya akses bahan baku, kepemilikan paten, lokasi yang tidak menguntungkan, serangan balik oleh perusahaan yang bertahan, dan potensi kejenuhan pasar.

Walaupun banyak hambatan, perusahaan baru kadang-kadang masuk ke dalam industri dengan produk yang lebih tinggi umumnya, harga yang lebih rendah, dan tenaga pemasaran yang banyak. Oleh karena itu, tugas perencana strategi adalah mengidentifikasi perusahaan baru yang potensial masuk pasar, memonitor strategi perusahaan baru yang menjadi pesaing, melakukan "serangan balasan" jika di perlakukan, dan memanfaatkan kekuatan dan kelemahan yang dimiliki.

- **Potensi Pengembangan Produk Pengganti**

Dalam berbagai industri, perusahaan bersaing dengan produsen pengganti. Contohnya, produsen tempat dari plastik bersaing dengan produsen tempat dari gelas, karton dan aluminium. Produsen asetaminofen bersaing dengan produsen obat sakit kepala yang lain. Adanya produk pengganti membuat batasan harga maksimal, sebelum konsumen pindah ke produk pengganti tersebut.

Tekanan persaingan akibat adanya produk pengganti semakin bertambah ketika harga produk pengganti relatif murah dan biaya konsumen untuk beralih ke produk pun rendah. Kekuatan kompetitif produk pengganti paling mudah diukur dari seberapa besar pangsa pasar yang direbutnya dan rencana perusahaan produk pengganti tersebut untuk meningkatkan kapasitas serta penetrasi pasar.

- **Kekuatan Tawar Pemasok**

Kekuatan tawar pemasok mempengaruhi intensitas persaingan dalam suatu industri, terutama ketika jumlah pemasok banyak, ketika hanya ada sedikit bahan baku pengganti yang baik, atau ketika biaya mengganti bahan baku amat tinggi. Sering kali demi kepentingan bersama, pemasok dan produsen saling membantu dengan memberikan harga yang terjangkau, mutu yang lebih baik, pengembangan pelayanan baru, penyerahan barang tepat waktu, dan mengurangi biaya inventarisasi, sehingga meningkatkan kemampuan meraih laba jangka panjang bagi semua pihak yang terkait.

Perusahaan mungkin menjalankan *backward integration strategy* atau strategi tarik mundur agar bisa mengendalikan pemasok atau menarik modal yang diberikan kepada pemasok. Strategi ini sangat efektif ketika pemasok tidak dapat diandalkan, biayanya terlalu tinggi, atau tidak mampu memenuhi kebutuhan perusahaan secara konsisten. Perusahaan biasanya dapat melakukan negosiasi persyaratan yang lebih menguntungkan dengan pemasok jika strategi ini lazim digunakan di antara perusahaan yang bersaing dalam industri.

- **Kekuatan Tawar Konsumen**

Ketika pelanggan terkonsentrasi atau jumlahnya besar, atau membeli dalam jumlah banyak, kekuatan tawarnya merupakan kekuatan utama yang mempengaruhi intensitas persaingan dalam suatu industri. Perusahaan pesaing mungkin menawarkan garansi yang lebih panjang atau pelayanan khusus untuk memperoleh loyalitas pelanggan ketika kekuatan tawar dari konsumen luar biasa. Kekuatan tawar konsumen juga lebih besar ketika produk yang dibeli bersifat standar atau tidak berbeda. Ketika demikian halnya, konsumen sering dapat melakukan negosiasi atau menekan harga jual, jaminan dan paket aksesori sampai tingkat tertentu.

2.7.2 Analisis SWOT

Analisis SWOT menurut Rangkuti (2004, p18) adalah identifikasi dari berbagai faktor secara sistematis untuk merumuskan strategi perusahaan. Pada dasarnya analisis SWOT terbagi atas dua bagian yaitu: analisis lingkungan internal dan eksternal. Analisis lingkungan internal terdiri dari kekuatan

(*strengths*) dan kelemahan (*weaknesses*), sedangkan lingkungan eksternal terdiri dari kesempatan (*opportunities*) dan ancaman (*threats*).

Berikut ini penjelasan mengenai faktor-faktor tersebut:

1. Kekuatan (*strengths*) : meliputi segala hal yang menjadi kekuatan utama perusahaan dibandingkan dengan pesaingnya. Misalnya: sumber daya, teknologi, SDM, modal, pengalaman, keterampilan, keunggulan persaingan, dan tingkat penguasaan pasar.
2. Kelemahan (*weaknesses*) : meliputi segala hal yang menjadi kelemahan perusahaan dibandingkan dengan pesaingnya. Misalnya: keterbatasan sumber daya, modal, pengalaman, kreativitas, dan kapabilitas yang menghambat kinerja perusahaan.
3. Kesempatan (*opportunities*) : meliputi segala hal yang dapat menjadi kesempatan atau situasi penting yang dapat menguntungkan perusahaan di dalam proses bisnisnya.
4. Ancaman (*threats*) : meliputi segala hal yang merupakan situasi yang kurang atau tidak menguntungkan bagi perusahaan dalam bisnisnya.

2.7.2.1 Matriks Evaluasi Faktor External

Menurut David (2004, p161), Matriks Evaluasi Faktor External (EFE) membuat perencana strategi dapat meringkas dan mengevaluasi informasi ekonomi, sosial, budaya, demografi, lingkungan, politik, pemerintah, hukum, teknologi dan persaingan.

Terdapat lima langkah dalam pengembangan matriks EFE.

1. Buat daftar faktor-faktor eksternal yang diidentifikasi dalam proses audit eksternal. Cari antara 10 dan 20 faktor, termasuk peluang-peluang dan

ancaman yang mempengaruhi perusahaan dan industrinya. Daftar peluang dahulu kemudian ancaman. Usahakan sespesifik mungkin, gunakan selalu presentase, rasio, dan angka perbandingan jika dimungkinkan.

2. Beri bobot pada setiap faktor dari 0,0 (tidak penting) sampai 1,0 (amat penting). Bobot menunjukkan kepentingan relatif dari faktor tersebut agar berhasil dalam industry tersebut. Peluang sering mendapat bobot lebih besar ketimbang ancaman. Tetapi ancaman dapat juga menerima bobot tinggi, jika berat atau sangat mengancam. Bobot yang wajar dapat ditentukan dengan membandingkan pesaing yang sukses dan yang gagal atau dengan mendiskusikan faktor tersebut dan mencapai konsensus kelompok. Jumlah seluruh bobot yang diberikan pada faktor diatas harus sama dengan 1,0.
3. Berikan peringkat 1 sampai 4 kepada masing-masing faktor eksternal kunci untuk menunjukkan seberapa efektif strategi perusahaan saat itu merespon faktor tersebut, dengan catatan: 4=respon luar biasa, 3=respon diatas rata-rata, 1=respon jelek. Peringkat didasarkan pada efektivitas strategi perusahaan. Peringkat didasarkan atas keadaan perusahaan, sedangkan bobot dalam langkah 2 didasarkan pada industry. Penting untuk diperhatikan bahwa baik peluang maupun ancaman dapat memperoleh peringkat 1, 2, 3, atau 4.
4. Kalikan setiap bobot faktor dengan peringkat untuk menentukan nilai yang dibobot.

5. Jumlahkan nilai yg dibobot untuk setiap variabel untuk menentukan nilai bobot total bagi organisasi.

   Berapa pun jumlah peluang dan ancaman utama yang dimasukkan dalam matriks EFE, total nilai yang dibobot tertinggi untuk suatu organisasi adalah 4,0 dan yang terendah adalah 1,0. Rata-rata nilai yang dibobot adalah 2,5. Jumlah nilai yang dibobot sama dengan 4,0 menunjukkan bahwa suatu organisasi memberi respon yang sangat bagus terhadap peluang-peluang dan ancaman yang ada dalam industrinya. Dengan kata lain, strategi perusahaan secara efektif memanfaatkan peluang yang ada dan meminimalkan potensi pengaruh negatif dari ancaman eksternal. Jumlah nilai yang dibobot sama dengan 1,0 menunjukkan bahwa strategi perusahaan tidak memanfaatkan peluang atau menghindari ancaman eksternal.

2.7.2.2 Matriks Evaluasi Faktor Internal

Menurut David (2004, p217), Langkah ringkas dalam melakukan audit manajemen strategis adalah membuat *Matriks Evaluasi Faktor Internal* atau IFE (Internal Factor Matriks). Alat perumusan strategi ini meringkas dan mengevaluasi kekuatan dan kelemahan utama dalam berbagai bidang fungsional dalam suatu usaha. Matriks ini juga menjadi landasan untuk mengidentifikasi dan mengevaluasi hubungan di antara bidang-bidang ini. Penilaian intuitif diperlukan dalam membuat matriks IFE. Karena itu, pendekatan yang tampaknya ilmiah ini janganlah dianggap sebagai teknik yang sangat ampuh. Pemahaman mendalam mengenai faktor-faktor yang dimuat dalam matriks ini lebih penting daripada sekedar angka-angka.

Matriks IFE dapat dikembangkan dalam lima langkah sebagai berikut :

1. Identifikasi faktor internal kunci yang mencakup baik kekuatan maupun kelemahan pada perusahaan.
2. Beri bobot masing-masing faktor dalam kelom kedua, mulai dari 0,0 (tidak penting) sampai dengan 1,0 (terpenting). Pembobotan menunjukkan tingkat kepentingan relatif dari faktor tersebut di dalam lingkup industri perusahaan. Tanpa memperhatikan jenis faktor, baik itu kekuatan maupun kelemahan, faktor yang dipertimbangkan memiliki efek paling besar dalam kinerja organisasi harus diberikan bobot yang paling tinggi. Jumlah dari semua bobot harus sama dengan 1.0.
3. Berikan peringkat 1 sampai dengan 4 pada setiap faktor internal kunci untuk mengindikasikan apakah faktor tersebut merepresentasikan kelemahan besar (peringkat=1), kelemahan kecil (peringkat=2), kekuatan kecil (peringkat=3), kekuatan besar (peringkat=4). Perlu diperhatikan bahwa kekuatan harus menerima peringkat 4 atau 3 dan kelemahan harus menerima peringkat 2 atau 1.
4. Kalikan setiap pembobotan faktor dengan peringkat untuk menentukan nilai.
5. Jumlahkan nilai untuk setiap variabel untuk menentukan total nilai yang dibobot untuk organisasi.

Berapa pun banyaknya faktor yang dimasukkan dalam Matris IFE, jumlah nilai yang dibobot dapat berkisar 1,0 yang rendah sampai 4,0 yang tinggi, dengan rata-rata 2,5. total nilai yang dibobot jauh dibawah 2,5 merupakan ciri organisasi yang lemah secara internal. Sedangkan jumlah

yang jauh diatas 2,5 menunjukkan posisi internal yang kuat. Jumlah bobot selalu berjumlah 1,0.

Ketika sebuah faktor internal utama merupakan kekuatan dan kelemahan, faktor itu harus dimasukkan dua kali dalam matriks IFE, dan bobot serta peringkat harus diberikan untuk setiap pernyataan.

2.4.2.3 Matrik TOWS

Matrik *Threats-Opportunities-Weakness-Strength* (TOWS) menurut David (2004, p288) merupakan perangkat pencocokan yang penting membantu manajer mengembangkan empat tipe strategi strategi: Strategi SO (*Strength-Opportunities*), Strategi WO (*Weakness-Opportunities*), Strategi ST (*Strength-Threats*), dan Strategi WT (*Weakness-Threats*). Mencocokan faktor-faktor eksternal dan internal kunci merupakan bagian yang sangat sulit dalam mengembangkan matriks TOWS dan memerlukan penilaian yang baik-dan tidak ada sekumpulan kecocokan yang paling baik.

Strategi SO atau strategi kekuatan-peluang menggunakan kekuatan internal perusahaan untuk memanfaatkan peluang eksternal. Semua manajer menginginkan organisasi mereka berada dalam posis di mana kekuatan internal dapat dipakai untuk memanfaatkan tren dan peristiwa eksternal. Organisasi umumnya akan menjalankan strategi WO, ST atau WT supaya mereka dapat masuk ke dalam situasi di mana mereka dapat menerapkan strategi SO. Jika perusahaan mempunyai kelemahan besar, perusahaan akan berusaha keras untuk mengatasinya dan membuatnya menjadi kekuatan. Kalau menghadapi ancaman besar, sebuah organisasi akan berusaha menghindarinya agar dapat memusatkan perhatian pada peluang.

Strategi WO atau strategi kelemahan-peluang bertujuan untuk memperbaiki kelemahan dengan memanfaatkan peluang eksternal. Kadang-kadang peluang eksternal yang besar ada, tetapi kelemahan internal sebuah perusahaan membuatnya tidak mampu memanfaatkan peluang itu.

Strategi ST atau strategi kekuatan-ancaman menggunakan kekuatan perusahaan untuk menghindari atau mengurangi dampak ancaman eksternal. Hal ini tidak berarti bahwa organisasi yang kuat pasti selalu menghadapi ancaman frontal dalam lingkungan eksternal.

Strategi WT atau strategi kelemahan-ancaman merupakan taktik defensif yang diarahkan utnuk mengurangi kelemahan internal dan menghindari ancaman eksternal. Sebuah organisasi yang dihadapkan pada berbagai ancaman eksternal dan kelemahan internal, sesungguhnya dalam posisi yang berbahaya. Faktanya, perusahaan seperti itu mungkin harus berjuang agar dapat bertahan atau melakukan merger, rasionalisasi, menyatakan pailit atau memilih dilikudasi.

Delapan langkah yang diperlukan untuk menyusun Matriks SWOT:

1. Tulis peluang eksternal kunci perusahaan.
2. Tulis ancaman eksternal kunci perusahaan.
3. Tulis kekuatan internal kunci perusahaan.
4. Tulis kelemahan internal kunci perusahaan.
5. Cocokkan kekuatan internal dengan peluang eksternal dan catatlah Strategi SO dalam sel yang sudah ditentukan.
6. Cocokkan kelemahan internal dengan peluang eksternal dan catatlah Strategi WO dalam sel yang sudah ditentukan.

7. Cocokkan kekuatan internal dengan ancaman eksternal dan catatlah Strategi ST dalam sel yang sudah ditentukan.
8. Cocokkan kekuatan internal dengan ancaman eksternal dan catatlah Strategi WT dalam sel yang sudah ditentukan.

Selalu dibiarkan kosong	KEKUATAN-S - - - Daftar Kekuatan - -	KELEMAHAN-W - - - Daftar Kelemahan - -
PELUANG-O - - - Daftar Peluang - -	STRATEGI SO - - Gunakan kekuatan untuk memanfaatkan peluang - - -	STRATEGI WO - - Atasi kelemahan dengan memanfaatkan peluang - - -
ANCAMAN-T - - - Daftar Ancaman -	STRATEGI ST - - Gunakan kekuatan untuk menghindari ancaman	STRATEGI WT - -Meminimalkan kelemahan dan menghindari ancaman

-	- - -	- - -

Tabel 2.5 Matriks TOWS
David (2004, p290)

2.8 Strategi Alternatif

Strategi alternatif yang dapat diambil oleh perusahaan adalah sebagai berikut:

STRATEGI	DEFINISI
Integrasi ke depan *(forward integration)*	Memiliki atau meningkatkan kendali atas distributor atau pengecer
Integrasi ke belakang *(backward integration)*	Mencoba memiliki atau meningkatkan kendali atas perusahaan pemasok
Integrasi Horizontal *(horizontal integration)*	Mencoba memiliki atau meningkatkan kendali atas para pesaing
Penetrasi Pasar *(market penetration)*	Mencari pangsa pasar yang lebih besar untuk produk atau jasa yang sudah ada sekarang melalui usaha pemasaran yang lebih gencar
Pengembangan Pasar *(product development)*	Memperkenalkan produk atau jasa yang sudah ada ke wilayah geografi baru
Pengembangan Produk *(product development)*	Mencoba meningkatkan penjualan dengan memperbaiki produk atau jasa yang sudah ada atau mengembangkan yang baru
Diversifikasi Konsentrik	Menambah produk atau jasa baru, tetapi masih

(concentric diversfication)	terkait
Diversifikasi Konglomerat *(conglomerate diversfication)*	Menambah produk atau jasa baru, yang tidak terkait, untuk para pelanggan baru
Diversifikasi Horisontal *(horizontal diversfication)*	Menambah produk atau jasa baru, yang tidak terkait, untuk pelanggan yang sudah ada
Rasionalisasi Biaya *(retrenchment)*	Merestrukturisasi dengan cara mengurangi biaya dan asset agar bisa meningkatkan penjualan dan keuntungan
Divestasi *(Divestiture)*	Menjual suatu divisi atau bagian dari suatu organisasi
Likuidasi *(Liquidation)*	Menjual semua aset sebuah perusahaan secara bertahap sesuai dengan nilainya yang terlihat

Tabel 2.6 Strategi Alternatif

David (2004, p233)