# BAB 2

# LANDASAN TEORI

## 2.1 Pengertian Sistem Informasi

Menurut O'Brien (2003, p7), penulis menterjemahkan bahwa sistem informasi merupakan kombinasi teratur dari orang – orang, *hardware*, *software*, jaringan komunikasi, dan sumber daya data yang mengumpulkan, mengubah, dan menyebarkan informasi dalam sebuah organisasi.

Menurut Laudon dan Laudon (2004, p8), *an information system can be defined technically as a set of interrelated components that collects (or retreive), process, store, and distribute information to support decision making and control in an organizatiton.*

Dari kedua definisi di atas, dapat ditarik kesimpulan bahwa sistem informasi merupakan unsur – unsur yang terdiri dari manusia, *hardware*, *software* dan jaringan yang diorganisasikan untuk memproses data (mengumpulkan, mengubah, menyebar) menjadi informasi untuk mencapai tujuan tertentu dalam perusahaan.

## 2.2 Pengertian Sistem Informasi Akuntansi

### 2.2.1 Pengertian Sistem Akuntansi

Menurut Bodnar dan Hopwood yang diterjemahkan oleh Amir Abadi Jusuf (2000, h181), sistem akuntansi adalah suatu organisasi yang terdiri dari metode dan catatan – catatan yang dibuat untuk mengidentifikasikan, mengumpulkan, menganalisis, mencatat, dan melaporkan transaksi - transaksi organisasi dan menyelenggarakan pertanggungjawaban bagi aktiva dan kewajiban yang berkaitan.

Menurut Wilkinson *et al* (2000, p5), menterjemahkan akuntansi memiliki beberapa sisi. Pertama, adalah suatu sistem informasi dalam cakupannya sendiri. Yaitu, sistem yang memakai berbagai macam operasi sistematis untuk membuat informasi yang relevan. Di antara operasi – operasi tersebut adalah (1) melakukan pencatatan data ekonomis (pengumpulan data), (2) melakukan pemeliharan data yang disimpan (pemeliharaan data), dan (3) menyajikan informasi kuantitatif dalam istilah – istilah finansial (pembuatan informasi).

Kesimpulannya sistem akuntansi merupakan suatu organisasi dalam perusahaan yang terdiri dari elemen tertentu untuk menyediakan informasi akuntansi yang dibutuhkan oleh manajemen untuk memudahkan pengelolaan perusahaan.

### 2.2.2 Pengertian Sistem Informasi Akuntansi

Menurut Bodnar dan Hopwood yang diterjemahkan oleh Amir Abadi Jusuf (2000, p6), sistem informasi akuntansi adalah sebagai sistem yang berbasis komputer yang dirancang untuk mengubah data akuntansi menjadi informasi, tetapi istilah sistem informasi akuntansi diperluas mencakup siklus – siklus pemrosesan transaksi, penggunaan teknologi informasi dan pengembangan sistem informasi.

Menurut Wilkinson *et al* (2000, p7), *accounting information system is a unified structure wihtin an entity, such as a business firm, that employs physical resources and other components to transform economic data into accounting information, with the purpose with satisfying the information needs of a variety of users.*

Dapat disimpulkan bahwa sistem informasi akuntansi merupakan suatu sistem yang mengolah data akuntansi menjadi informasi yang dibutuhkan untuk merencanakan, mengendalikan, dan mengoperasikan bisnis.

### 2.2.3 Tujuan dan Kegunaan Sistem Informasi Akuntansi

Menurut Wilkinson *et al* (2000, p8), sistem informasi akuntansi memiliki tujuan sebagai berikut :

- sebagai pendukung kegiatan operasional sehari – hari perusahaan.
- sebagai pendukung pihak manajemen dalam membuat keputusan.
- sebagai suatu acuan oleh pihak eksternal untuk mengetahui keadaan perusahaan.

Menurut Jones dan Rama (2006, p6), yang diterjemahkan oleh penulis, sistem informasi akuntansi memiliki 5 kegunaan, yaitu :

- Perusahaan menggunakan sistem informasi akuntansi untuk membuat laporan mengenai informasi yang dibutuhkan oleh pihak eksternal perusahaan seperti *investor,* kreditur, dan penagih pajak.
- Manajer menggunakan sistem informasi akuntansi dalam menangani aktivitas operasional yang rutin dilakukan dalam suatu siklus operasi suatu perusahaan seperti menerima pesanan pelanggan, pemenuhan jasa, pengiriman barang, menagih dan menerima pembayaran.
- Informasi dibutuhkan untuk mendukung pengambilan keputusan pada semua tingkat manajemen dalam perusahaan, baik yang dilakukan secara rutin maupun *ad hoc*.
- Sistem informasi juga dibutuhkan untuk membantu dalam membuat suatu perencanaan dan juga dalam melakukan pengendalian atas setiap aktivitas yang dilakukan.

- Melaksanakan pengendalian internal, dimana termasuk aturan – aturan, prosedur, dan menjaga keakuratan data keuangan perusahaan.

### 2.2.4 Komponen – Komponen Sistem Informasi Akuntansi

Menurut Romney dan Steimbart (2006, p6), terdapat 6 komponen dari sistem informasi akuntansi, yaitu:

1. Orang yang mengoperasikan sistem dan melaksanakan berbagai macam fungsi.
2. Prosedur – prosedur dan instruksi – instruksi , baik manual maupun terotomatisasi, yang terlibat dalam pengumpulan, pemrosesan, dan penyimpanan data mengenai aktivitas – aktivitas organisasi.
3. Data mengenai organisasi dan proses bisnisnya.
4. *Software* yang digunakan untuk memproses data organisasi.
5. Infrastruktur teknologi informasi, yang mencakup komputer - komputer, perangkat pendukung, dan perangkat komunikasi jaringan yang digunakan untuk mengumpulkan, menyimpan, memproses, dan mengirimkan data dan informasi.
6. Pengendalian internal dan pengukuran keamanan yang mengamankan data dalam sistem informasi akuntansi.

### 2.2.5 Siklus Sistem Informasi Akuntansi

Menurut Wilkinson *et al* (2000, p45), siklus sistem informasi akuntansi yang merupakan siklus transaksi akuntansi (*transaction cycles*) terdiri dari:

***General Ledger and Financial Reporting Cycle***

Merupakan pusat dari siklus lainnya. Siklus ini unik di mana pemrosesan transaksi individual bukanlah merupakan fungsi keseluruhannya maupun fungsinya yang penting. Selain itu, juga lebih banyak bekerja sama dengan pemrosesan yang berhubungan dengan akuntansi daripada kejadian bisnis. Arus masuk utamanya timbul dari *output* siklus transaksi lainnya. Sebagai tambahan, siklus ini meliputi transaksi non rutin dan penyesuaian yang timbul selama atau pada akhir tiap periode akuntansi.

***Revenue Cycle***

Siklus ini meliputi tiga kejadian bisnis atau transaksi kunci: permintaan atas proyek, eksekusi proyek dan pengiriman (penjualan), serta penerimaan kas.

***Expenditure Cycle***

Siklus ini meliputi dua kejadian bisnis atau transaksi kunci: pembelian dan pengeluaran kas.

***Resources-Management Cycle***

Siklus ini terdiri dari semua aktivitas yang berhubungan dengan sumber daya fisik perusahaan. Jadi melibatkan kejadian bisnis sebagai berikut:

- Memperoleh modal dari berbagai sumber (termasuk pemilik), menginvestasikan modal dan membayar modal ke penerimanya.
- Memperoleh, memelihara, dan menyingkirkan fasilitas (aset tetap).
- Memperoleh, menyimpan, dan menjual persediaan (barang dagangan).
- Memperoleh, memelihara, dan membayar personil (seperti pegawai, manajer, konsultan dan pihak luar lainnya).

***Other Transaction Cycles***

Siklus ini merupakan siklus-siklus lain selain yang telah dijelaskan di atas, yang tergantung dari jenis perusahaan. Misalnya pada perusahaan manufaktur menambahkan siklus produksi atau konversi (*production/conversion cycle*).

## 2.3 Sistem Informasi Akuntansi Pembelian

Menurut Assauri (2008, h224), pembelian adalah merupakan fungsi staf yang kedudukannya setingkat dengan jabatan – jabatan senior lainnya seperti *controller* dan manajer penjualan, karena mempunyai tanggung jawab yang besar terhadap keuangan dan kelancaran operasi perusahaan.

Menurut Bodnar dan Hopwood (2000, h277) yang diterjemahkan oleh Amir Abadi Jusuf, pembelian adalah kegiatan yang dilakukan oleh perusahaan dengan membeli barang secara tunai atau kredit atau membeli aktiva produksi untuk digunakan dalam kegiatan perusahaan atau membeli barang dan jasa berhubungan dengan kegiatan perusahaan.

Dapat disimpulkan bahwa pembelian merupakan suatu kegiatan dalam perusahaan yang berhubungan dengan fungsi/bagian dalam perusahan ataupun kemampuan perusahaan untuk membeli suatu barang atau jasa agar dapat digunakan dalam aktivitas perusahaan.

### 2.3.1 Fungsi – Fungsi yang Terkait

Menurut Wilkinson *et al* (2000, 470-471), fungsi - fungsi yang terkait dalam siklus pengeluaran (*expenditure cycle)* adalah:

***Inventory Management/Logistic***

Dalam suatu perusahaan dagang, tujuan dari fungsi *inventory management* adalah untuk mengelola persediaan dagang yang didapatkan oleh perusahaan untuk dijual kembali. Dalam perusahaan manufaktur, aktivitas yang terlibat dalam *inventory management* mungkin dikombinasikan dengan produksi untuk membentuk fungsi logistik yang lebih luas. Selain bertanggung jawab atas perencanaan, *inventory management* juga mencakup pembelian, penerimaan, dan penyimpanan.

Bagian pembelian berfokus pada pemilihan *vendor* atau *supplier* yang paling tepat untuk memesan barang atau jasa. Hal ini membuat pemilihan dilakukan berdasarkan beberapa faktor seperti harga unit barang atau jasa, kualitas barang atau jasa yang ditawarkan, syarat dan tanggal pengiriman yang dijanjikan, dan kehandalan dari *supplier*. Bersama dengan pengendalian persediaan (yang berada di bawah fungsi *accounting*), bagian pembelian akan menjamin kuantitas dari barang yang didapatkan. Bagian penerimaan memiliki tanggung jawab untuk menerima hanya pada barang – barang yang dipesan, memverifikasi kuantitas dan kondisi barang tersebut, dan memindahkan barang ke gudang. Bagian gudang memiliki tanggung jawab untuk menjaga barang dari pencurian, kehilangan, dan perusakan serta menyiapkan dengan tepat waktu ketika terdapat permintaan atas barang tersebut.

***Finance/Accounting***

Tujuan dari *financing* dan *accounting management* berhubungan dengan pembiayaan, data, informasi, perencanaan, dan pengendalian terhadap sumber daya. Dalam hubungannya dengan siklus pembelian, tujuan dibatasi hanya untuk pengendalian dan perencanaan kas, untuk mengatur data yang berkaitan dengan pembelian dan akun

*supplier*, untuk pengendalian persediaan, dan untuk mengatur informasi yang berkaitan dengan kas, pembelian, dan *supplier*.

### 2.3.2 Tugas dan Tanggung Jawab Bagian Pembelian

Menurut Assauri (2008, h228-229), tanggung jawab bagian pembelian berbeda – beda di setiap perusahaan pada luasnya aktivitas yang dilakukan dan dipengaruhi oleh operasi yang ekonomis dari perusahaan tersebut. Adapun tanggung jawab bagian pembelian antara lain:

1. Bertanggung jawab atas pelaksanaan pembelian bahan – bahan agar rencana operasi dapat dipenuhi dan pembelian bahan – bahan tersebut pada tingkat harga yang perusahaan pabrik akan mampu bersaing dalam memasarkan produknya.
2. Bertanggung jawab atas usaha – usaha untuk dapat mengikuti perkembangan bahan – bahan baru yang dapat menguntungkan dalam proses produksi, perkembangan dalam desain, harga dan faktor – faktor lain yang dapat mempengaruhi produksi perusahaan, harga dan desainnya.
3. Bertanggung jawab untuk meminimalisasi investasi atau meningkatkan perputaran (*turn over*) bahan, yaitu dengan penentuan skedul arus bahan ke dalam pabrik dalam jumlah yang cukup untuk memenuhi kebutuhan produksi.
4. Bertanggung jawab atas kegiatan penelitian dengan menyelidiki data dan perkembangan pasar, perbedaan sumber – sumber penawaran (*supply*) dan memeriksa pabrik *supplier* untuk mengetahui kapasitasnya dan kemampuan untuk memenuhi kebutuhan – kebutuhan perusahaan.

5. Sebagai tambahan, kadang – kadang bertanggung jawab atas pemeliharaan bahan – bahan yang dibeli setelah diterima, yaitu pekerjaan – pekerjaan di gudang pabrik dan bertanggung jawab atas pengawasan persediaan (*inventory control*).

Tugas - tugas yang dilakukan bagian pembelian dalam memenuhi tanggung jawab antara lain:

1. Melakukan pembelian bahan – bahan secara bersaing atas dasar nilai yang ditentukan tidak hanya oleh harga yang tepat tetapi juga oleh waktu yang tepat, jumlah dan mutu/kualitas yang tepat.
2. Membantu melakukan pemlihan bahan – bahan dengan menyelidiki/ substitusi.
3. Untuk memperoleh sumber – sumber pilihan dari suplai dengan melakukan usaha – usaha pencarian paling sedikit dua sumber dari suplai.
4. Mempengaruhi tingkat persediaan terendah (*the lowest stock levels)*
5. Menjaga hubungan dengan *supplier* yang baik
6. Melakukan kerja sama dan koordinasi yang efektif dengan fungsi – fungsi lainnya dalam perusahaan.
7. Melakukan penelitian tentang keadaan perdagangan dan pasar.
8. Melakukan pembelian seluruh bahan – bahan dan perlengkapan yang dibutuhkan. tepat pada waktunya sehingga tidak mengganggu rencana produksi dari perusahaan pabrik tersebut.

### 2.3.3 Prosedur dalam Sistem Akuntansi Pembelian

Menurut Bodnar dan Hopwood (2004) prosedur dalam proses pembelian adalah:

1. Menentukan kebutuhan
2. Memilih sumber daya

3. Permintaan untuk *quotation*
4. Memilih pemasok
5. Menerbitkan pesanan pembelian
6. Penerimaan barang
7. Pembayaran kepada pemasok

### 2.3.4 Dokumen Sistem Akuntansi Pembelian

Menurut Wikinson (2000, p472) dokumen yang terkait pada sistem akuntansi pembelian, yaitu:

- ***Purchase Requisition***

  *Form* yang digunakan dalam siklus pembelian yang mengotorisasi penempatan dari *order* untuk barang atau jasa.

- ***Purchase Order***

  *Form* yang secara resmi disiapkan yang berasal dari permintaan pembelian.

- ***Receiving Order***

  Dokumen yang mencatat penerimaan barang.

- ***Supplier (Vendor) Invoice***

  *Invoice* dari *supplier* yang menyediakan barang atau jasa.

- ***Disbursment Voucher***

  Dokumen di dalam sistem *voucher* yang mengakumulasikan *supplier invoice* untuk pembayaran.

- ***Disburstment Check***

  Dokumen terakhir dalam siklus pembelian yang menyediakan pembayaran kepada *supplier* untuk suatu barang atau jasa.

- ***Debit Memorandum***

  Dokumen yang mengotorisasi pengembalian pembelian.

- ***New Supplier Form***

  *Form* yang digunakan dalam pemilihan *supplier* baru, menunjukkan data mengenai harga, tipe barang atau jasa yang disediakan, pengalaman, posisi kredit, referensi.

- ***Request for proposal (or quotation)***

  *Form* yang digunakan salam prosedur pemilihan, menunjukkan barang atau jasa yang diperlukan dan harga yang bersaing, jangka waktu pembayaran dan lain – lain.

## 2.4 Sistem Informasi Akuntansi Persediaan

Menurut Assauri (2008, h237), persediaan adalah suatu aktiva yang meliputi barang – barang milik perusahaan dengan maksud untuk dijual dalam suatu periode usaha yang normal, atau persediaan barang – barang yang masih dalam pengerjaan atau proses produksi, ataupun persediaan bahan baku yang menunggu penggunanya dalam proses produksi.

Menurut Sumayang (2003, h197), *inventory* atau persediaan merupakan simpanan material yang berupa bahan mentah, barang dalam proses dan barang jadi.

Dari sudut pandang sebuah perusahaan maka persediaan adalah sebuah investasi modal yang dibutuhkan untuk menyimpan material pada kondisi tertentu.

Dari pengertian diatas dapat disimpulkan bahwa persediaan merupakan suatu aktiva yang dapat dibagi menjadi beberapa bagian (bahan baku, bahan setengah jadi, barang jadi) yang penggunaannya tergantung dari sisi kebutuhan perusahaan yang dapat digunakan untuk membuat suatu produk atau untuk dijual.

### 2.4.1 Fungsi Persediaan

Menurut Render dan Heizer (2005, h60), persediaan dapat melayani beberapa fungsi yang akan menambahkan fleksibilitas operasi perusahaan. Empat fungsi persediaan adalah:

1. Untuk men-"*decouple*"atau memisahkan beragam bagian proses produksi. Sebagai contoh jika pasokan sebuah perusahaan berfluktuasi, maka mungkin diperlukan persediaan tambahan untuk men-*decouple* proses produksi dari para pemasok.
2. Untuk men-*decouple* perusahaan dari fluktuasi permintaan dan menyediakan persediaan barang – barang yang akan memberikan pilihan bagi pelanggan. Persediaan semacam ini umumnya terjadi pada pelanggan eceran.
3. Untuk mengambil keuntungan diskon kuantitas, sebab pembelian dalam jumlah lebih besar dapat mengurangi biaya produksi atau pengiriman barang.
4. Untuk menjaga pengaruh inflasi dan naiknya harga.

### 2.4.2 Manfaat Persediaan

Menurut Ridwan (2002, h260-261), manfaat memiliki persediaan bagi perusahaan adalah:

**1. Menghindari kehilangan penjualan**

Jika perusahaan tidak mempunyai barang yang tersedia untuk dijual maka perusahaan dapat kehilangan penjualan. Pelanggan mungkin akan membeli dari pesaing atau mungkin pelanggan yang tidak mau menunggu tidak akan membeli dari dari perusahaan. Kemampuan perusahaan untuk memberikan pelayanan yang cepat dan ketepatan pengiriman barang sangat tergantung pada manajemen persediaan yang baik.

**2. Memperoleh diskon kuantiti**

Jika perusahaan ingin mempunyai persediaan yang besar untuk suatu produk tertentu maka perusahaan dimungkinkan untuk membeli barang dalam jumlah besar guna memperoleh diskon kuantiti. Dengan membeli lebih murah, maka perusahaan dapat meningkatkan laba sepanjang biaya pengadaan persediaan lebih kecil dari diskon yang diperoleh.

**3. Mengurangi biaya persediaan**

Setiap kali menempatkan pesanan untuk persediaan, perusahaan akan mengeluarkan sejumlah biaya sehubungan dengan memiliki persediaan tersebut. Pekerjaan administrasi yang akan dilakukan sehubungan dengan adanya pesanan antara lain formulir harus diperiksa, disetujui, dan dikirimkan. Ketika barang tiba, barang harus diterima, diperiksa dan dihitung untuk kemudian disimpan digudang. Faktur harus dicocokkan dengan barangnya dan dikirim ke bagian akunting sehingga *supplier* dapat dibayar. Biaya variabel yang berkaitan dengan

pesanan dapat dikurangi jika frekuensi pesanan dikurangi. Biaya yang berkaitan dengan gudang seperti membuat gudang, biaya pemeliharaan dan perbaikan, biaya gaji SDM dan lainnya dapat dikurangi bila perusahaan memiliki persediaan yang tidak berlebihan.

**4. Mencapai biaya produksi yang efisien**

Biaya penyetelan mesin akan terjadi sebelum produksi dimulai. Sebagai contoh, jika biaya penyetelan mesin Rp2.000.000 dan diproduksi 200.000 unit maka biaya per unit Rp. 10. jika diproduksi 2.000.000 unit maka biaya per unit Rp1. Secara jangka panjang persediaan dapat membuat perusahaan mencapai produksi yang efisien. Persediaan bahan baku yang cukup juga mengurangi kemungkinan kekurangan barang yang dapat menunda atau mengganggu produksi.

### 2.4.3 Jenis – Jenis Persediaan

Menurut Assauri (2008, h240), persediaan yang terdapat dalam perusahaan dapat dibedakan menurut beberapa cara. Dilihat dari fungsinya, persediaan dapat dibedakan atas:

***Batch Stock atau Lot Size Inventory***

Persediaan yang diadakan karena kita membeli atau karena kita membeli atau membuat bahan – bahan/ barang – barang dalam jumlah yang lebih besar dari jumlah yang dibutuhkan pada saat itu.

***Fluctuation Stock***

Persediaan yang diadakan untuk menghadapi fluktuasi permintaan konsumen yang tidak dapat diramalkan.

***Anticipation Stock***

Persediaan yang diadakan untuk menghadapi fluktuasi permintaan yang dapat diramalkan, berdasarkan pola musiman yang terdapat dalam satu tahun dan untuk menghadapi penggunaan atau penjualan permintaan yang meningkat.

Di samping perbedaan menurut fungsi, persediaan dapat dibedakan atau dikelompokkan menurut jenis dan posisi barang tersebut didalam urutan pengerjaan produk, yaitu:

- Persediaan bahan baku (*Raw Material Stock*), yaitu persediaan barang – barang berwujud yang digunakan dalam proses produksi, barang mana dapat diperoleh dari sumber – sumber alam ataupun dibeli dari *supplier* atau perusahaan yang menghasilkan bahan baku bagi perusahaan pabrik yang menggunakannya.
- Persediaan bagian produk atau *part* yang dibeli (*purchased parts/komponents stock*), yaitu persediaan barang – barang yang terdiri atas *part* yang diterima dari perusahaan lain yang dapat secara langsung di-*assembling* dengan *part* lain, tanpa melalui proses produksi sebelumnya.
- Persediaan bahan – bahan pembantu atau barang – barang perlengkapan (*supplies stock*) yaitu persediaan barang – barang atau bahan – bahan yang diperlukan dalam proses produksi atau yang dipergunakan dalam bekerjanya suatu perusahaan, tetapi tidak merupakan bagian atau komponen dari barang jadi.
- Persediaan barang setengah jadi atau barang dalam proses (*work in process/progress stock*) yaitu persedian barang – brang yang keluar dari tiap – tiap bagian dalam suatu pabrik atau bahan – bahan yang telah diolah menjadi suatu bentuk, tetapi perlu lebih diproses kembali untuk kemudian menjadi barang jadi.

- Persediaan barang jadi (*finished good stock*) yaitu barang - barang yang telah selesai diproses atau diolah dalam pabrik dan siap untuk dijual kepada pelanggan atau perusahaan lain.

### 2.4.4 Dokumen yang Digunakan dalam Persediaan

Menurut Assauri (2008, h283), pencatatan dalam pengawasan persediaan adalah semua pencatatan atau pembukuan mengenai penerimaan, persediaan di gudang dan pengeluaran bahan baku dan bahan – bahan lainnya serta hasil produksi dalam suatu perusahaan. Pencatatan – pencatatan tersebut diperlukan untuk menjamin bahan – bahan atau barang – barang dipergunakan secara efisien dan perusahaan dapat mengikuti perkembangan persediaannya dengan baik.

Menurut Assauri (2008, h286), pada dasarnya terdapat lima catatan yang paling penting atau utama dalam sistem pengawasan persediaan:

**Permintaan untuk dibeli ( *Purchase requisition* )**

Dokumen ini merupakan permintaan dari sebagian persediaan kepada bagian pembelian untuk membeli bahan – bahan / barang – barang yang sesuai dengan jenis dan jumlah tertentu seperti yang dinyatakan dalam surat permintaan itu.

**Laporan Penerimaan ( *Receiving report* )**

Dokumen ini penting karena satu *copy*/rangkap dari laporan ini akan memberikan informasi bahwa penjaga gudang akan telah menerima bahan – bahan yang dipesan ini dipabrik.

**Daftar Persediaan ( *Balance of store forms* )**

Dokumen ini merupakan catatan yang paling penting dalam pengawasan persediaan. Dokumen ini merupakan dasar atau titik pangkal dari pelaksanaan sistem

pengawasan persediaan dan memberikan informasi baik bagi pabrik maupun bagi bagian accounting.

Informasi yang terdapat dalam " *balances of stores card* " berbeda – beda tergantung dari perusahaan pabrik yang menggunakannya. Akan tetapi data-data yang biasanya terdapat dalam daftar ini adalah gambaran atau deskripsi lengkap dari bahan – bahan tersebut

- jumlah dari bahan – bahan yang tersedia di gudang, yang dipesan dan dialokasikan untuk produksi.
- jumlah bahan – bahan yang ada atau harus dibeli bila waktunya telah tiba untuk mengadakan pemesanan baru.
- harga bahan – bahan per unit .
- jumlah yang dipakai selama suatu periode atau jangka waktu tertentu.
- nilai dari persediaan yang ada.

**Formulir permintaan bahan (*Material requisition form*)**

Formulir yang dibuat oleh bagian gudang untuk dipergunakan oleh bagian pembelian dalam mengadakan pesanan. Daftar ini juga penting dalam pengawasan persediaan karena dapat menunjukkan bahan – bahan yang perlu segera dibeli untuk pengisian kembali persediaan gudang.

**Perkiraan Pengawasan (*Control accounting*)**

*Material control accounting* umumnya untuk menjaga supaya perkiraan (*general ledger*) yang dibuat oleh bagian akuntansi tetap merupakan alat yang penting dalam sistem pengawasan yang efektif. Semua pembelian akan didebit dan semua pemakaian

akan dikredit dalam perkiraan ini, sehingga saldonya harus sama dengan saldo yang terdapat pada " *perpetual inventory cards* ".

### 2.4.5 Model Persediaan

Menurut Render dan Heizer (2005, h67), model pengendalian persediaan menganggap bahwa permintaan untuk sebuah barang mungkin bebas (*independent)* atau terikat (*dependent)* dengan permintaan barang lain. Bagaimana pun, permintaan untuk komponen pemanggang roti *terikat* dengan kebutuhan pemanggang roti.

Menurut Indrajit dan Djokopranoto (2003, h221), pada dasarnya ada dua jenis permintaan, yaitu permintaan *independent* (bebas) dan permintaan *dependent* (tergantung / tidak bebas). Permintaan *dependent* timbul apabila kebutuhan dipicu oleh suatu kejadian spesifik. Dalam pabrik, kejadian spesifik ini berupa keperluan suatu rakitan (*assembly)* untuk menjadi sebuah produk. Permintaan *independent* timbul apabila kebutuhan barang tersebut tidak berhubungan dengan barang lain atau kejadian tertentu.

### 2.4.6 Metode Pencatatan Persediaan

Menurut Assauri (2008, h244), ada 2 sistem yang umum dikenal dalam menentukan jumlan persediaan pada akhir suatu periode, yaitu dengan:

1. ***Periodic System,*** yaitu setiap akhir periode dilakukan perhitungan secara fisik dalam menentukan jumlah persediaan akhir.
2. ***Perpetual System*** atau juga disebut dengan *Book Inventories* yaitu dalam hal ini dibina catatan administrasi persediaan. Setiap mutasi dari persediaan sebagai akibat dari pembelian ataupun penjualan dicatat atau dilihat dalam Kartu Administrasi

persediaannya. Bila metode ini yang dipakai, maka perhitungan secara fisik hanya dilakukan paling tidak setahun sekali yang biasanya dilakukan untuk keperluan *counterchecking* antara jumlah persediaan menurut fisik dengan menurut catatan dalam Kartu Administrasi Persediaannya.

### 2.4.7 Metode Penilaian Persediaan

Menurut Assauri (2008, h246), dalam menilai persediaan ada beberapa cara yang dapat digunakan, di antaranya dengan:

1. Cara *First-In, First-Out* (FIFO)
2. Cara rata – rata ditimbang (*weighted averaged method*)
3. Cara *Last-In, First-Out* (LIFO)

**1. Cara *First-In, First-Out* (FIFO)**

Cara ini didasarkan atas asumsi bahwa harga barang yang sudah terjual dinilai menurut harga pembelian barang yang terdahulu masuk. Dengan demikian persediaan akhir dinilai menurut harga pembelian barang yang akhir masuk.

**2. Cara rata – rata ditimbang (*weigthed averaged method*)**

Cara ini berbeda dengan cara yang dijelaskan sebelumnya karena didasarkan atas harga rata – rata di mana harga tersebut dipengaruhi oleh jumlah barang yang diperoleh pada masing – masing harganya.

**3. Cara *Last-In, First-Out* (LIFO)**

Cara ini didasarkan atas asumsi bahwa barang yang telah terjual dinilai menurut harga pembelian barang yang terakhir masuk. Sehingga persediaan yang masih ada/*stock*, dinilai berdasarkan harga pembelian barang yang terdahulu.

### 2.4.8 Sistem Persediaan Bahan Baku

#### 2.4.8.1 ROP (*Reorder Point*)

Sumayang (2003, h211-212), *re-order point* adalah posisi persediaan yang ditentukan sebagai batas untuk melakukan pemesanan ulang. *re-order point* ditetapkan pada tingkat persediaan yang cukup tinggi untuk mengurangi resiko kemungkinan persediaan habis dan untuk menghitung kemungkinan ini, perlu diketahui data statistik tentang pola penyebaran permintaan selama tenggang waktu pemesanan atau *lead time* tersebut.

Menurut Indrajit dan Djokopranoto (2003, p53), waktu pemesanan atau kapan memesan adalah setiap kali persediaan mencapai titik minimum. Titik di mana pemesanan ini dilakukan dinamakan titik pemesanan kembali (*reorder point*)

Penghitungan *re-order point* sebagai berikut :

$$ROP = d \times L + ss$$

ROP = titik pemesanan ulang

d = pemakaian bahan baku per hari

L = *Lead Time,* waktu tunggu

ss = *safety stock/*persediaan pengaman

Contoh perhitungan ROP:

Diketahui: jumlah hari kerja per tahun = 288 hari

d = demand = $\frac{1800 \text{ ton}}{288 \text{ hari}}$ = 6,25 ton

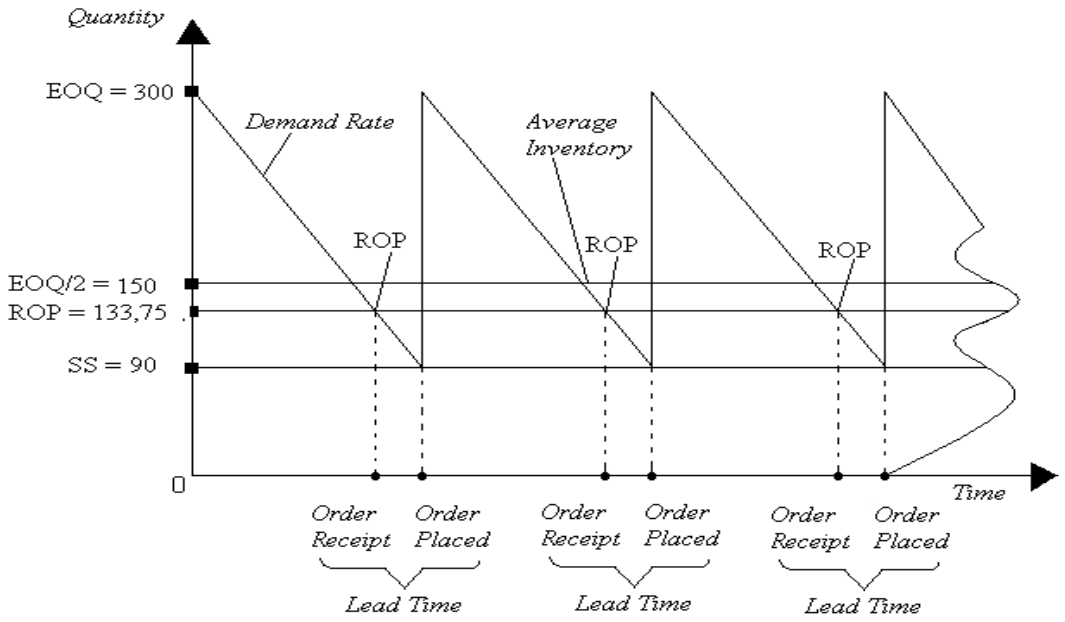
Tenggang waktu pemesanan (*lead time)* = 7 hari

*safety stock* (ss) = 5% dari total permintaan per tahun

= 5% x 1800 ton = 90 ton

ROP = d x L + ss

= (6,25 x 7) + 90 = 133,75 ton




Gambar 2.1 Grafik ROP

### 2.4.9 *Safety Stock*

Menurut Asssauri (2008, h263), persediaan penyelamat (*safety stock*) adalah persediaan tambahan yang diadakan untuk melindungi atau menjaga kemungkinan terjadinya kekurangan bahan (*stock-out*).

Menurut Indrajit dan Djokopranoto (2003, h171), persediaan pengaman (*safety stock*) adalah persediaan ekstra yang harus diadakan untuk proteksi atau pengaman dalam menghindari kehabisan persediaan karena berbagai sebab. Dengan demikian persediaan pengaman mempunyai dua aspek dalam pembiayaan perusahaan, yaitu:

1. Persediaan pengaman akan mengurangi biaya yang timbul karena kehabisan persediaan. Makin besar pengaman, makin kecil kemungkinan

kehabisan persediaan, sehingga makin kecil pula biaya karena kehabisan persediaan.

2. Tetapi adanya persediaan pengaman akan menambah biaya penyediaan barang. Makin besar persediaan pengaman, makin besar pula biaya penyediaan barang.

Dari kedua teori di atas dapat disimpulan bahwa *safety stock* adalah persediaan cadangan yang kuantitasnya melebihi kebutuhan persediaan akan permintaan dalam periode tertentu yang dilakukan perusahaan untuk dapat menghindari kehabisan persediaan dikarenakan berbagai macam kondisi seperti gagal produksi, keterlambatan pengiriman bahan baku, bencana alam, dan lainnya.

**Penentuan *Safety Stock***

Menurut Sumayang (2003, 210 - h213), *safety stock* dapat ditentukan berdasarkan pada dua unsur, yaitu *service level* dan standar deviasi permintaan selama *lead time.*

Rumus *safety stock :*

*Safety stock* (S) = zs

z = ditentukan oleh *service level*

s = standar deviasi perimntaan selama *lead time*

Tabel 2.1 Persentase Permintaan Normal

| *NORMAL DEMAND PERCENTAGES* | | |
|---|---|---|
| z | *Service Level, Percent* | *Stockout, Percent* |
| 0 | 50,0 | 50,0 |
| 0,5 | 69,1 | 30,9 |
| 1,0 | 84,1 | 15,9 |
| 1,1 | 86,4 | 13,6 |
| 1,2 | 88,5 | 11,5 |
| 1,3 | 90,3 | 9,7 |
| 1,4 | 91,9 | 8,1 |
| 1,5 | 93,3 | 6,7 |
| 1,6 | 94,5 | 5,5 |
| 1,7 | 95,5 | 4,5 |
| 1,8 | 96,4 | 3,6 |
| 1,9 | 97,1 | 2,9 |
| 2,0 | 97,7 | 2,3 |
| 2,1 | 98,2 | 1,8 |
| 2,2 | 98,6 | 1,4 |
| 2,3 | 98,9 | 1,1 |
| 2,4 | 99,2 | 0,8 |
| 2,5 | 99,4 | 0,6 |
| 2,6 | 99,6 | 0,5 |
| 2,7 | 99,6 | 0,4 |
| 2,8 | 99,7 | 0,3 |
| 2,9 | 99,8 | 0,2 |
| 3,0 | 99,9 | 0,1 |

*Sumber : Dasar – Dasar Manajemen Produksi & Operasi, Lalu Sumayang, Salemba Empat*

Contoh perhitungan:

Perusahaan "A" adalah pengelola gudang bahan makanan yang memasok tepung terigu ke para pengecer dengan karakteristik sebagai berikut:

*Service level* = 95%

*Lead time* = 4 hari

Standar deviasi dari permintaan = 150 unit per hari

Maka:

Standar deviasi permintaan selama *lead time* 4 hari adalah $s^2 = 4 \times 150^2$ atau $s = \sqrt{4 (150^2)}$, s = 300 unit.

95% *service level* maka z = 1,65 (lihat tabel 2.2)

*Safety stock* (S) = zs

= 1,65 x 300 unit = **495 unit**

### 2.4.10 Tingkat Pelayanan atau *Service level*

Menurut Sumayang (2003, h210), *service level* merupakan istilah yang banyak digunakan dalam manajemen persediaan yang merupakan besar presentase dari permintaan pelanggan yang dapat terpenuhi dari persediaan. Maka 100% *service level* berarti semua permintaan pelanggan dapat dipenuhi dari persediaan, sehingga dengan demikian :

*Stockout* = 100 – *service level*

Ada beberapa cara untuk menjelaskan arti *service level* yaitu :

1. *Service level* adalah sebuah kemungkinan di mana suatu permintaan pelanggan dapat dipenuhi dari persediaan selama tenggang waktu pemesanan atau *lead time* dalam satu siklus pemesanan.
2. *Service level* adalah besar persentase permintaan yang dapat dipenuhi dari persediaan dalam periode waktu tertentu.
3. *Service level* adalah besar persentase dari waktu berapa lama inventori mempunyai persediaan.

## 2.5 Sistem Pengendalian Internal

Menurut Wilkinson *et al.* (2000,h150), pengendalian internal sebagai sebuah sistem, struktur, atau proses yang diimplematasikan oleh direksi perusahaan, manejemen dan anggota lainnya, yang diharapkan mampu memberikan jaminan tentang pencapaian tujuan dari pengendalian dalam kategori berikut ini:

- Efektivitas dan efeisensi operasi – operasi
- Keandalan laporan keuangan
- Kepatuhan dengan hukum dan peraturan yang berlaku

Menurut Bodnar dan Hopwood (2001, p182), pengendalian internal adalah sebuah proses yang dipengaruhi oleh para dewan direksi dari sebuah perusahaan, manajemen dan ponsel lain yang dirancang untuk menyediakan keyakinan yang masuk akal berkaitan dengan pencapaian tujuan dalam kategori: kehandalan pelaporan keuangan, efektivitas dan efesiensi operasional, dan kepatuhan akan hukum dan peraturan yang berlaku.

Dari definisi di atas dapat ditarik kesimpulan bahwa pengendalian internal merupakan sistem yang digunakan untuk meningkatkan perfoma dan profitabilitas perusahaan serta meminimalisir penyimpangan – penyimpangan yang terjadi dalam perusahaan.

### 2.5.1 Tujuan Pengendalian Internal

Menurut Romney dan Steinbart (2006, p96), tujuan pengendalian internal adalah sebagai berikut :

1. Menjaga *asset,* termasuk mencegah atau mendeteksi, secara regular, perolehan, penggunaan, atau pembuangan material yang tidak terotorisasi dari *asset* perubahan.
2. Memelihara catatan dalam detail yang cukup untuk secara akurat dan sesuai menggambarkan *asset* perusahaan.
3. Menyediakan informasi yang akurat dan dapat dipercaya.
4. Menyediakan kepastian yang masuk akal bahwa pelaporan keuangan dipersiapkan sesuai dengan GAAP (*Generally Accepted Accounting Principles*).
5. Mempromosikan dan meningkatkan efisiensi operasional, termasuk memastikan penerimaan dan pengeluaran perusahaan dibuat sesuai dengan otorisasi manajer dan direktur.
6. Menggalakkan keterikatan pada kebijakan manajerial yang telah dirumuskan.
7. Menyesuaikan dengan hukum dan peraturan yang ditetapkan.

### 2.5.2 Komponen Pengendalian Internal

Menurut Jones dan Rama (2006, h105), komponen-komponen yang berhubungan dengan pengendalian internal terdiri dari lima komponen yaitu:

**1. *Control Environment***

Berhubungan dengan beberapa faktor yang disusun organisasi untuk mengontrol kesadaran para karyawannya. Faktor tersebut berhubungan dengan integritas, nilai etika, filosofi manajemen dan *operating style.* Hal ini juga termasuk cara manajemen menetapkan otoritas dan tanggung jawab, mengatur, dan mengembangkan sumber daya manusia serta perhatian dan petunjuk dari *board of directors*.

**2. *Risk Assessment***

Merupakan proses identifikasi dan analisis terhadap risiko yang dapat menghambat pencapaian tujuan pengendalian internal.

**3. *Control Activities***

- Merupakan kebijakan dan prosedur yang dikembangkan oleh organisasi untuk menangani risiko-risiko yang mungkin dan telah ada. *Control activities* mencakup:
- *Performance reviews*, kegiatan yang berhubungan dengan analisis terhadap kinerja, misalnya dengan membandingkan hasil yang didapat dengan anggaran, standar perhitungan, dan data pada periode sebelumnya.
- *Segregation duties*, terdiri dari penetapan tanggung jawab untuk mengotorisasi transaksi, melakukan transaksi, mencatat transaksi, dan menjaga aset yang dilakukan oleh karyawan yang berbeda.
- *Application control*, berhubungan dengan aplikasi SIA.
- *General control*, berhubungan dengan pengawasan yang lebih luas yang berhubungan dengan berbagai aplikasi.

**4. *Information and Communication***

Sistem informasi perusahaan adalah kumpulan dari prosedur (baik otomatis maupun manual) dan pencatatan dalam memulai, mencatat, memproses, dan melaporkan kejadian atas proses-proses yang terjadi dalam organisasi, dan komunikasi berhubungan dengan menyediakan pemahaman atas peraturan dan tanggung jawab individu.

**5. *Monitoring***

Manajemen harus mengawasi pengendalian internal untuk memastikan bahwa pengendalian internal organisasi berjalan sesuai tujuan yang ditetapkan.

### 2.5.3 Pengendalian Internal Sistem Akuntansi Pembelian

Menurut Jones dan Rama (2006, p442-445), pengendalian internal pada siklus pembelian meliputi:

1. Pemisahan tugas. Individu – individu yang mengotorisasi, melaksanakan pembelian, dan mencatat transaksi adalah individu yang berbeda untuk menghindai terjadinya kecurangan.
2. Menggunakan informasi dari kejadian lampau untuk mengontrol aktivitas pembelian.
3. Mengamati dari dekat semua kegiatan pembelian.
4. Dokumen – dokumen yang berurutan dan bernomor urut tercetak.
5. Mencatat semua pihak yang bertanggung jawab atas proses yang terjadi.
6. Membatasi akses ke aset dan informasi perusahaan.
7. Merekonsilidasi semua catatan dengan bukti fisik dari aset yang ada.

### 2.5.4 Pengendalian Internal Sistem Persediaan

Menurut Brigham dan Houston (2006, h164-167) sistem pengendalian persediaan terbagi menjadi:

1. Metode Garis Merah (*red-line method*)

   Suatu prosedur pengendalian persediaan di mana sebuah garis merah digambarkan di sekitar bagian dalam wadah penyimpanan persediaan sebagai indikasi tingkat titik pemesanan ulang

2. Metode Dua Wadah (*two-bin method*)

   Suatu prosedur pengendalian persediaan di mana pemesanan akan dilakukan ketka salah satu dari dua wadah yang menyimpan persediaan kosong.

3. Sistem Terkomputerisasi

   Suatu sistem pengendalian persediaan di mana komputer digunakan untuk menentukan titik pemesanan ulang dan untuk melakukan penyesuaian terhadap saldo persediaan.

4. Sistem *Just-In-Time*

   Suatu sistem pengendalian persediaan di mana perusahaan manufaktur mengkoordinasi produksinya dengan para pemasok sehingga bahan baku atau komponen diterima tepat pada saat mereka dibutuhkan di dalam proses produksi.

5. *Out – Sourcing*

   Suatu praktik pembelian komponen daripada membuatnya sendiri. *Out – sourcing* sering kali dikombinasikan dengan sistem *just-in-time* untuk mengurangi tingkat persediaan. Namun begitu salah satu alasan utama dari *out-sourcing* mungkin sama sekali tidak berhubungan dengan kebijakan persediaan.

### 2.5.5 Manajemen Data

Menurut Wilkinson *et al* (2000, p103), terdapat dua orientasi yang dapat digunakan dalam memodel data yaitu:

1. *File – Oriented Systems*

Dalam sistem *file – oriented,* setiap aplikasi memiliki jumlah *user* yang terbatas yang terlibat dalam pemrosesan data tipe khusus yang diperoleh dan menggunakan *output*nya untuk kebutuhan khususnya. Hirarki sistem ini terdiri dari *data element, record,*dan *file*

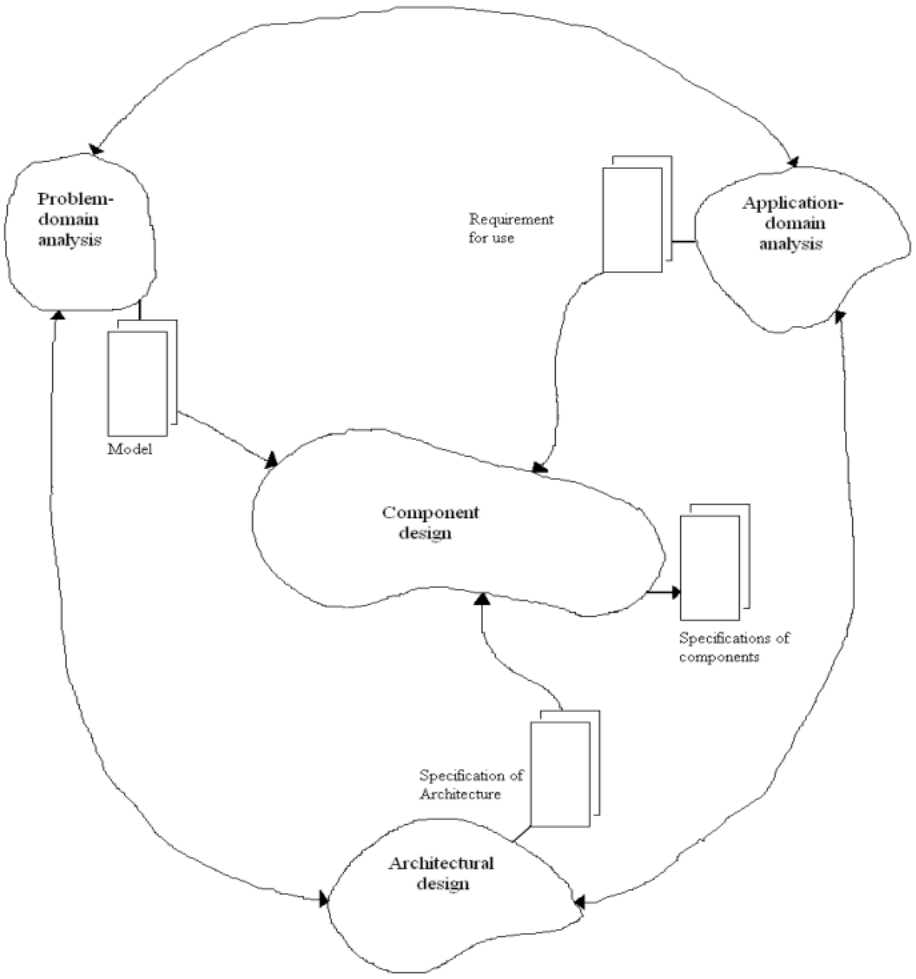
2. *Database Systems*

Dalam sistem *database,* data dipandang sebagai sumber daya pusat dan sama – sama digunakan untuk semua *user* yang berkepentingan dan aplikasinya. Fokusnya adalah data sebagai *asset.* Aplikasi yang memproses data diberikan lebih sedikit prioritas. Hirarki dari sistem ini adalah *data set* dan *database*

## 2.6 Pengertian Metode Analisis, Perancangan dan Desain Berorientasi Objek

Menurut Mathiassen *et al.* (2000, p4), OOA&D (*Object Oriented Analysis and Design)* menggunakan *object* dan *class* sebagai konsep utama dan membangun 4 prinsip umum untuk desain dan analisis: memodelkan *system context,* menekankan pertimbangan *architectural,* penggunaan kembali *pattern,* dan menghubungkan *method* di setiap situasi yang berkembang. Prinsip – prinsip ini adalah fondasi untuk OOA&D.

Menurut Mathiassen *et al.* (2000, p15), terdapat 4 aktivitas utama dalam OOA&D, yaitu *Problem Domain Analysis, Application Domain Analysis, Architectural Design,* dan *Component Design.*




Gambar 2.2 Aktivitas Utama OOA&D
*Sumber : Mathiassen et al, 2000, p15*

### 2.6.1 *Object* dan *Class*

Menurut Mathiassen *et al.* (2000, p4), *object* adalah suatu entitas dengan identitas, keadaan (tingkatan hidup) dan tingkah laku.Dalam analisis, objek merupakan perwujudan dalam *system's context,* seperti pelanggan. Sedangkan *class* adalah suatu deskripsi atas kumpulan objek yang saling menggunakan struktur, pola tingkah laku, dan atribut secara bersama-sama. *Class* berguna untuk memahami *object* dan sangat penting untuk mendeskripsikan mereka (*object).*

### 2.6.2 Prinsip – Prinsip Umum dalam Analisis, Perancangan dan Desain Berorientasi Objek

Menurut Mathiassen *et al.* (2000, p6-12), OOA&D membangun empat prinsip umum untuk analisis dan desain, yaitu :

- ***Model the Context***

  Konteks sistem dapat dilihat dari dua perspektif; *problem domain* dan *application domain.*

  *Problem domain :* bagian dari konteks yang diadministrasi, dimonitor atau dikendalikan oleh sistem

  *Application domain :* organisasi yang mengadministrasi, memonitor, atau mengendalikan *problem domain.*

- ***Emphasize the Architecture***

  *System:* adalah kumpulan dari komponen – komponen yang mengimplementasikan kebutuhan model, fungsi, dan tampilan. Arsitektur sistem harus mudah dipaham karena sistem ini menyediakan dasar keputusan dan sebagai komunikasi dan alat kerja untuk pengembangan. Sistem ini juga harus *flexible* karena pengembangan sistem berada pada lingkungan yang tidak stabil. Pada akhirnya, arsitektur sistem harus berguna karena keberhasilan sitem tergantung pada peran yang dimainkan dalam organisasi.

- ***Reuse Pattern***

  Suatu cara dasar untuk memastikan kualitas dan efesiensi dalam desain dan analisis adalah dengan menggunakan kembali ide – ide yang telah diuji dan

digunakan pada berbagai situasi. OOA&D mengilhami penggunaan kembali (*reuse*) ini dengan dua cara: dengan menggunakan komponen dan *object*, dan dengan menggunakan pola desain dan analisis.

- ***Tailor the Method***

  OOA&D adalah kumpulan dari petunjuk umum untuk desain dan analisis. Oleh karenanya harus disesuaikan dengan projek dan organisasi. Untuk membuat metode lebih berguna, kita harus mendesain metode agar adaptasi, peningkatan, dan penggantian bagian dapat lebih mudah diimplementasikan.

### 2.6.3 *System Choice*

Menurut Mathiassen *et al.* (2000, p25), s*ystem choice* didasarkan pada tiga subaktivitas. Subaktivitas pertama berfokus pada tantangan: kita mencoba untuk memperoleh gambaran baik dari situasi dan berbagai cara orang mengintepretasikannya. Subaktivitas kedua adalah menciptakan dan mengevaluasi ide – ide untuk desain sistem. Metode kita menawarkan sekumpulan tehnik untuk mendukung kreativitas dan memperkenalkan cara baru untuk berpikir. Dalam subaktivitas ketiga, kita memformulasikan dan memilih *system definition.* Mendiskusikan dan mengevaluasi alternatif *system definition* dalam hubungannya dengan berbagai macam situasi.

### 2.6.4 *System Definition*

Menurut Mathiassen *et al.* (2000, p24), *system definition* adalah gambaran ringkas dari sistem terkomputerisasi yang dinyatakan dalam bahasa alami. *System*

*deifinition* mengekspresikan sifat – sifat dasar dari pengembangan dan penggunaan sistem. Sistem ini menggambarkan sistem dalam konteks, apa yang seharusna terkandung dalam informasi, fungsi – fungsi apa yang harus disediakan, di mana harus digunakan, dan kondisi pengembangan yang bagaimana yang harus diterapkan.

Tujuan dari *system definition* itu sendiri adalah untuk menjelaskan secara lengkap perbedaan kemungkinan – kemungkinan dan intepretasi . Sistem ini membantu untuk memelihara *overview* dari pemilihan – pemilihan yang berbeda, dan dapat digunakan untuk membandingkan alternatif – alternatif. *System definition* yang dipilih harus menyediakan landasan – landsan yang baik untuk kelangsungan aktivitas desain dan analisis.

### 2.6.5 *Rich Picture*

Menurut Mathiassen *et al.* (2000, p26), *rich picture* adalah suatu gambaran informal yang mewakili pemahaman liustrator terhadap situasi dari suatu sistem. Dengan menggunakan *rich picture* kita dapat menjelaskan secara lengkap pandangan user terhadap situasi, permasalahan, dan mendapatkan pandangan terhadap suatu situasi dengan cepat. Tujuannya adalah tidak untuk menciptakan gambaran yang detail dari semua kemungkinan yang ada di lingkungan, tetap lebih untuk mendapatkan gambaran umum.

### 2.6.6 FACTOR *Criterion*

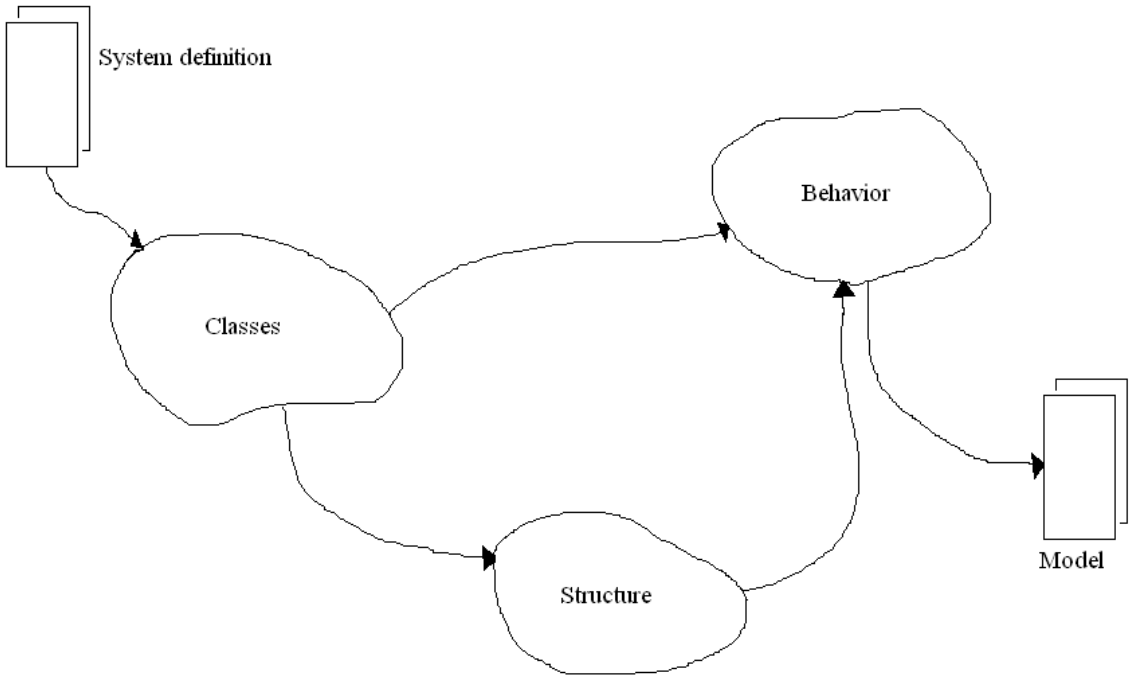
Menurut Mathiassen *et al.* (2000, p39-40), kriteria *FACTOR* terdiri dari enam elemen:

- *Functionality*: Fungsi sistem yang mendukung tugas-tugas *application domain*
- *Application Domain*: Bagian organisasi yang mengadministrasi, memonitor, dan mengendalikan *problem domain.*
- *Condition*: Kondisi dimana sistem akan dikembangkan dan digunakan.
- *Technology*: Mencakup teknologi yang akan digunakan untuk mengembangkan sistem dan teknologi dimana sistem akan dijalankan.
- *Objects*: Objek utama dari *problem domain*.
- *Responbility*: Tanggung jawab keseluruhan dari sistem dalam hubungannya dengan konteks.

### 2.6.7 *Problem-Domain Analysis*

Menurut Mathiassen *et al.* (2000), *Problem-domain analysis* adalah analisa terhadap sistem bisnis dalam dunia nyata yang dapat diatur, dimonitor, atau dikendalikan oleh sistem. Tujuan dari *problem-domain analysis* adalah untuk mengidentifikasikan dan

membuat model dari *problem domain.* Tujuan dari aktivitas ini adalah membangun sebuah model yang dapat digunakan untuk merancang dan mengimplementasikan sebuah sistem yang dapat memproses, berkomunikasi, dan menyajikan informasi mengenai *problem domain.* Hasil dari *problem domain analysis* adalah *class diagram*.

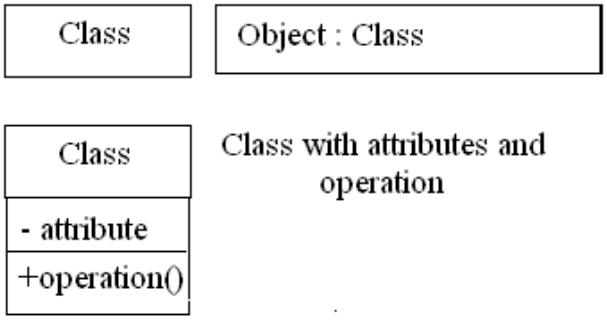



Gambar 2.3 Aktivitas dalam *Problem Domain Analysis*
*Sumber : Mathiassen et al, 2000, p46*

### 2.6.7.1 *Classes*

Menurut Mathiassen *et al.* (2000, p53), *Class* adalah gambaran atau definisi kumpulan objek yang berbagi *structure, behaviour pattern,* dan *attribute* yang bersamaan. *Class* merupakan kegiatan yang pertama dilakukan didalam analisis *problem-domain.* Pemilihan *class* bertujuan untuk mendefinisikan dan membatasi *problem domain,* sedangkan pemilihan *event* bertujuan untuk membedakan tiap – tiap *class* dalam *problem domain.*

Menurut Mathiassen *et al.* (2000, p51), *event* merupakan kejadian secara terus – menerus yang melibatkan satu atau lebih dari suatu *object.*



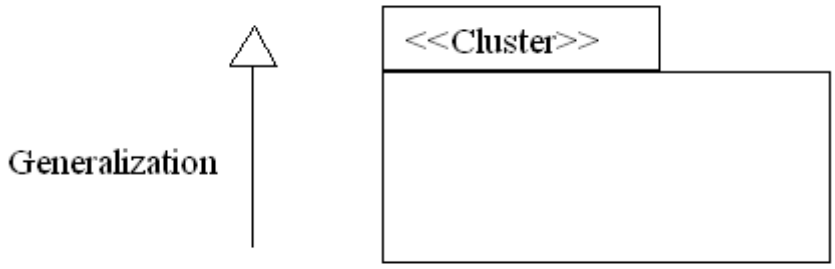

Gambar 2.4 Notasi Dasar dari *Class*
*Sumber : Mathiassen et al, 2000, p337-339*

### 2.6.7.2 *Structure*

Menurut Mathiassen *et al.* (2000, p336), *structure* adalah hubungan antara class dengan object pada problem domain secara keseluruhan. *Structure* bertujuan untuk menggambarkan hubungan terstruktur antara *classes* dan *object* dalam *problem domain*. Hasil dari *structure* adalah *class diagram*. *Class diagram* menggambarkan kumpulan dari *classes* dan hubungan yang terstruktur.

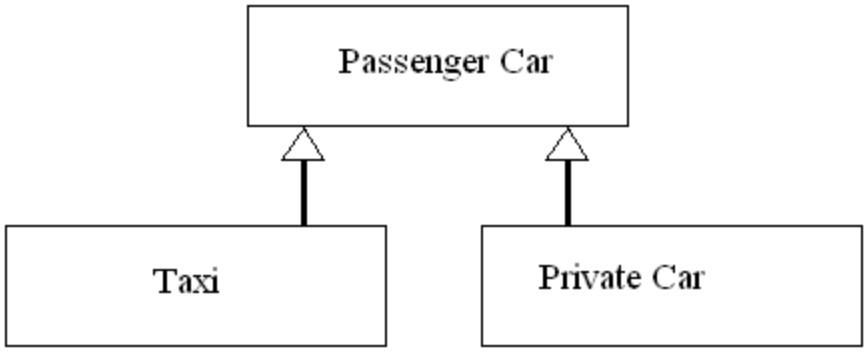
Menurut Mathiassen *et al.* (2000, p72-77), terdapat dua tipe *structure* dalam *obeject-oriented,* yaitu:

1. *Class structure,* mengekspresikan hubungan konseptual yang statis antar *class*. Hubungan statis ini tidak akan berubah, kecuali terjadi perubahan pada deskripsinya. *Class structure* terbagi menjadi dua macam, yaitu:




Gambar 2.5 Notasi *Class Structure*
*Sumber : Mathiassen et al, 2000, p337*

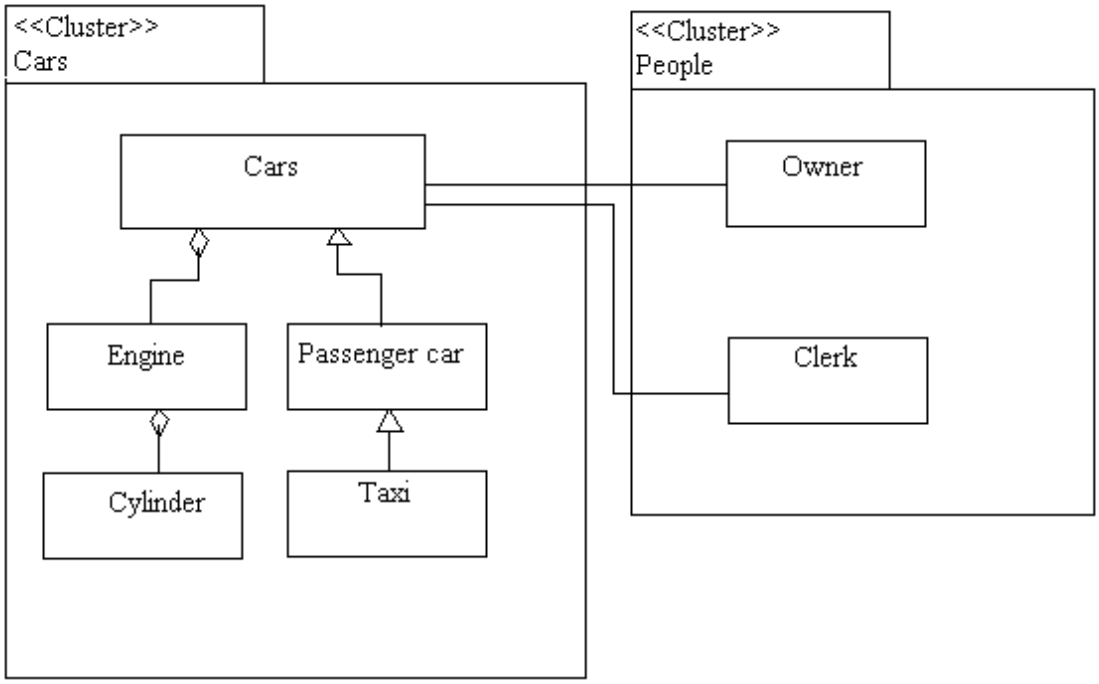
- ***Generalization Structure,*** merupakan hubungan antara dua atau lebih *subclass* yang umum (*superclass*). *Superclass* mendeskripsikan properti umum kepada *group* dari *special class* (*subclass*). Atau dengan kata lain, terjadi penurunan *attributes* dan *behaviour* dari *superclass*, tetapi *sublass* juga diperkenankan untuk memiliki *attributes* dan *behaviour* tambahan. Secara ilmu bahasa, *generalization structure* diekspresikan dengan formula "*is a*".




Gambar 2.6 *Generalization Structure*
*Sumber : Mathiassen et al, 2000, p73*

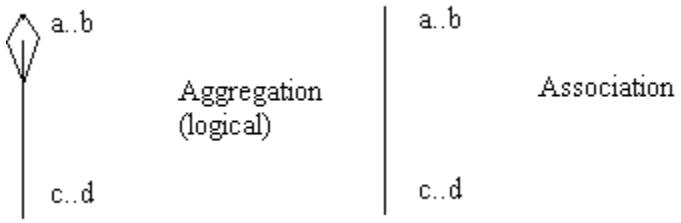
- ***Cluster,*** merupakan kumpulan dari *class* yang berhubungan . *Cluster* digambarkan dengan notasi *file folder* yang melingkupi *class – class* yang saling berhubungan di dalamnya. *Class – class* dalam satu *cluster* biasanya memiliki

hubungan berupa *generalization* atau *aggregation.* Sedangkan hubungan *class* dengan cluster yang berbeda biasanya berupa *association structure.*




Gambar 2.7 *Cluster Structure*
*Sumber : Mathiassen et al, 2000, p75*

2. *Object structure,* mengekpresikan hubungan dinamis antar *object.* Hubungan ini dapat berubah secara dinamis tanpa mempengaruhi perubahan pada deskripsinya. Biasanya terdapat *multiplicity* yang menspeifikasikan jumlah dari *object* yang berealisasi. *Multiplicity* dapat berupa *string of number* dan penyebaran internal dengan koma, seperti "0,3,7,9..,13,19,,*", "*" disebut *many,* dan 0...*. ada dua macam *object structure* yaitu:




Gambar 2.8 Notasi *Object Structure*
*Sumber : Mathiassen et al, 2000, p337*

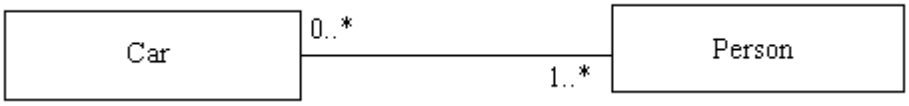
- ***Aggregation structure,*** mendefinisikan hubungan antara dua atau lebih *object.* Sebuah *superior object* (*whole*) memiliki beberapa *object* (*parts*). Secara ilmu bahasa, *aggregation structure* diekspresikan dengan formula "*has a*", "*a-part-of*", atau "*is-owned-by*". Menurut Mathiassen *et al.* (2000, p79), terdapat tiga tipe *aggregation structure,* yaitu:

  ***Whole part,*** di mana *whole* merupakan jumlah dari *parts,* sehingga jika salah satu *parts* dihilangkan maka secara tidak langsung mengubah *whole.*

  ***Container-content,*** di mana *whole* merupakan kontainer (tempat menampung) dari *parts*-nya, sehingga bila terdapat penambahan atau pengurangan terhadap isinya (*parts*), tidak akan mengubah pengertian *whole*-nya.

  ***Union-member,*** di mana *whole* merupakan *union*/gabungan yang terorganisir dari anggotanya (*parts*), sehingga jika terdapat penambahan atau pengurangan anggota, tidak akan mengubah *union*-nya. Terdapat batasan jumlah anggota terendah, karena tidak mungkin sebuah *union* tanpa anggota.

- ***Assosiation Structure,*** mendefinisikan hubungan antara dua atau lebih *object,* tetapi berbeda dengan *aggregation.* Hubungan antar *class – class* pada *aggregation* mempunyai pertalian yang kuat sedangkan pada *association* tidak kuat. Secara ilmu bahasa, *association structure* diekspresikan dengan formula "*knows*" atau "*associated-with*"




Gambar 2.9 *Association Structure*
*Sumber : Mathiassen et al, 2000, p77*

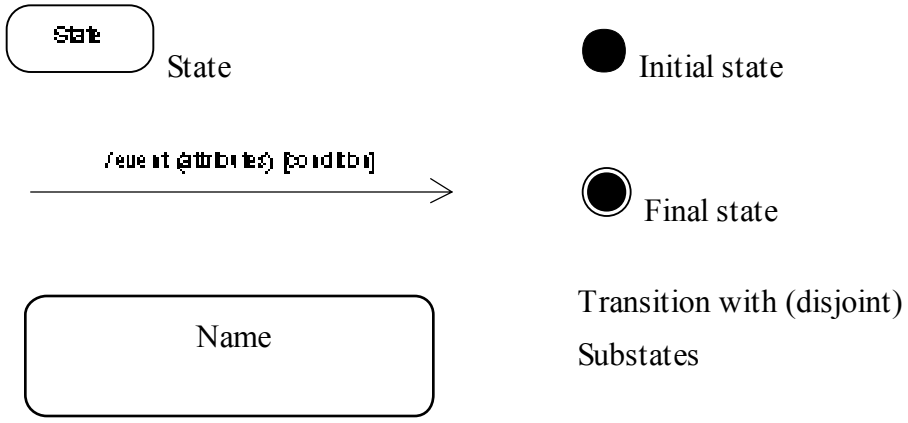
### 2.6.7.3 *Behavior*

Menurut Mathiassen *et al.* (2000, p93), kegiatan *behavior* bertujuan untuk memodelkan apa yang terjadi (perilaku dinamis) dalam *problem-domain* sistem sepanjang waktu. Tugas utama dari kegiatan ini adalah menggambarkan pola perilaku (*behavioral pattern*) dan atribut dari setiap kelas. Hasil dari kegiatan ini adalah *statechart diagram.*

Menurut Mathiassen *et al.* (2000, p89), *behavioral pattern* memiliki struktur pengendalian sebagai berikut:

- *Sequence* adalah suatu set *event* yang akan terjadi satu per satu (secara berurutan). Notasinya: "+".
- *Selection* adalah satu *event* yang terjadi dari suatu set *events.* Notasinya: "|".
- *Iteration* adalah satu *event* yang terjadi berulang – ulang kali. Notasinya: "*".

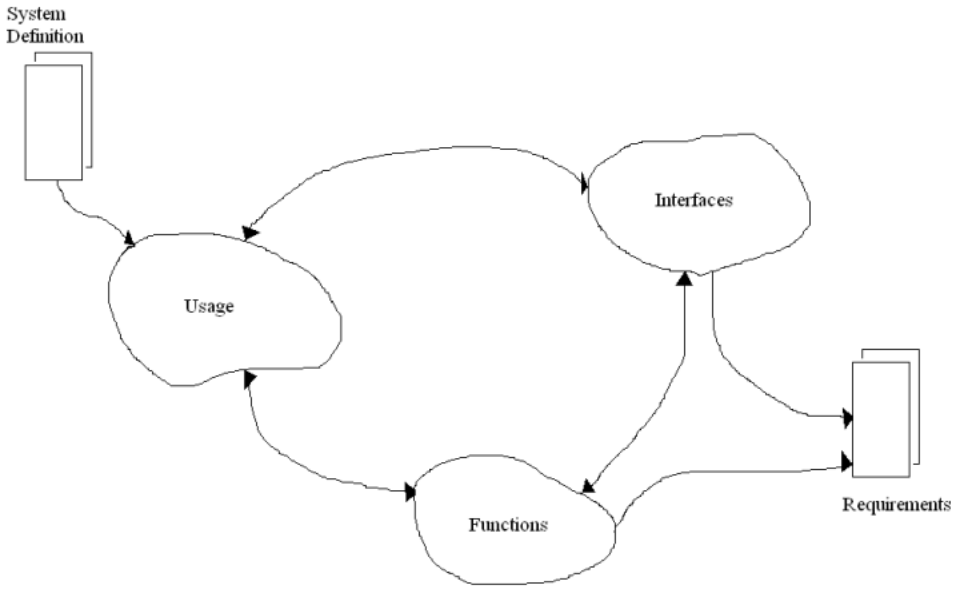
Jika menghadapi situasi *behavior pattern* yang kompleks, akan sulit sekali untuk engekspresikannya dalam notasi – notasi umum sehingga untuk pengekspresiannya lebih cenderung menggunakan *Statechart Diagram*.




Gambar 2.10 Notasi Dasar *Statechart Diagram*
*Sumber : Mathiassen et al, 2000, p341*

### 2.6.8 *Application Domain Analysis*

Menurut Mathiassen *et al.* (2000, p6), a*pplication domain* adalah suatu bagian yang mengatur, memonitor, atau mengendalikan *problem-domain. Application domain analysis* memfokuskan pada bagaimana target sistem akan digunakan dengan menentukan kebutuhan *function* dan *interface.*



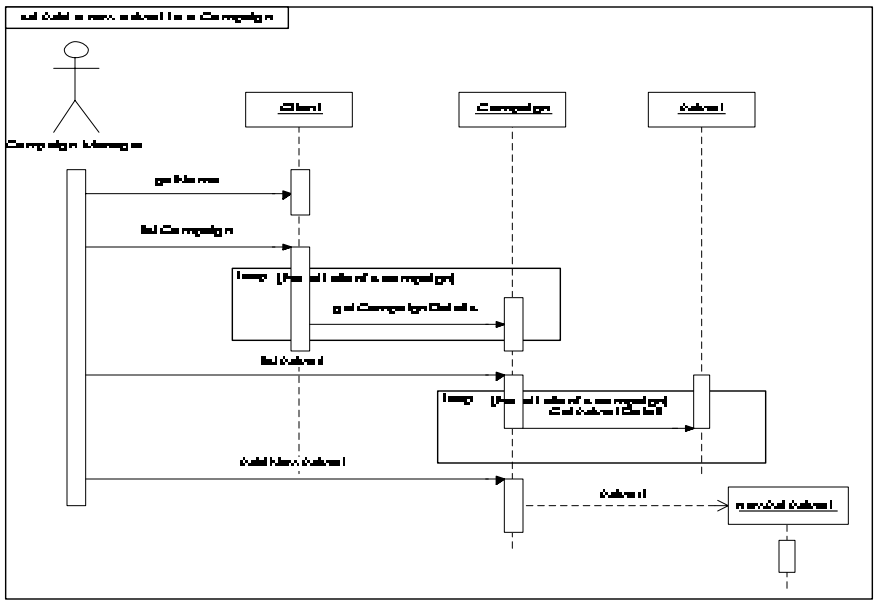

Gambar 2.11 Aktivitas dalam *Application Domain Analysis*
*Sumber : Mathiassen et al, 2000, p117*

#### 2.6.8.1 *Usage*

Mengacu pada pendapat Mathiassen *et al.* (2000), tujuan dari kegiatan *usage* adalah untuk menentukan bagaimana aktor-aktor yang merupakan pengguna atau sistem lain berinteraksi dengan sistem yang dituju. Interaksi antara *actor* dan sistem dinyatakan dalam *use case.* Menurut Mathiassen *et al.* (2000, p119-120), *use case* adalah pola interaksi antara sistem dan *actor* dalam *application domain.* Sedangkan *actor* adalah sebuah abstraksi dari pengguna atau sistem lain yang berinteraksi dengan target sistem. Hasil akhir dari aktivitas ini adalah membuat deskripsi dari *actor* dan *use case,* di mana relasinya diekspresikan dengan menggunakan *actor table* atau *use case diagram.*

### 2.6.8.2 *Sequence Diagram*

Menurut Benneth (2006, p329), menyatakan bahwa *sequence diagram* membantu seorang analis untuk mengidentifikasi pada tingkat detil operasi yang penting untuk mengimplementasikan fungsi dari *use case.*



Gambar 2.12 Contoh *Sequence Diagram* untuk "*Add a New Advert to a Campaign* "

Menurut Benneth (2006, p270), tabel 2.2 menunjukkan operator interaksi yang digunakan dalam *sequence diagram.*

Tabel 2.2 Operator Interaksi yang Mungkin Digunakan dengan Kombinasi Fragmen

| | |
|---|---|
| *Alt* | *Alternative* melambangkan pilihan *behaviours.* Setiap pilihan *behaviour* ditampilkan dalam *operand* terpisah. *Operand* yang merupakan interaksi pembatas dievaluasi sebagai pelaksanaan sebenarnya. |
| *Opt* | *Option* menggambarkan sebuah pilihan dari sebuah *operand* yang akan hanya dilaksanakan jika interaksi pembatas dievaluasi sebagai sebenarnya. |
| *Break* | *Break* mengindikasikan bahwa fragmen yang dikombinasikan dilakukan pada *remainder* dari fragmen interaksi penutupan. |

| | |
|---|---|
| *Par* | *Parallel* mengindikasikan bahwa pelaksanaan *operand* dalam fragmen yang dikombinasikan mungkin bergabung dalam *sequence* apapun ketika *event sequence* dalam setiap *operand* dipertahankan. |
| *Seq* | *Sequence* yang lemah menghasilkan setiap *operand* yang dipelihara tetapi kejadian *event* dari *operand* yang berbeda pada *lifeline* yang berbeda mungkin saja terjadi. |
| *Strict* | Peruntunan *strict* menentukan *strict sequence* dalam pelaksanaan dari *operand* tetapi tidak dipergunakan untuk sekumpulan fragmen. |
| *Neg* | *Negative* melambangkan *operand* yang tidak berlaku |
| *Critical* | Bagian *critical* memaksakan pembatas pada *operand* bukan kejadian *event*-nya pada *lifeline* dalam bagian yang dapat disisipkan. |
| *Ignore* | *Ignore* mengindikasikan jenis pesan, penetapan sebagai *parameter,* yang harus diabaikan pada interaksi. |
| *Consider* | *Consider* menetapkan pesan mana yang harus dipertimbangkan dalam interaksi. |
| *Assert* | *Assertion* menetapkan bahwa *sequence* dari suatu pesan dalam *operand* hanya yang berlaku terus – menerus. |
| *Loop* | *Loop* digunakan untuk mengindikasikan *operand* yang diulang berulang kali sampai pembatas interaksi untuk *loop* tidak lagi sebenarnya. |

#### 2.6.8.3 *Function*

Menurut Mathiassen *et al.* (2000, p138), *function* adalah sebuah fasilitas untuk membuat model berguna untuk *actors*. Adapun tujuan dari *function* adalah untuk menentukan kemampuan pemrosesan sistem informasi. *Function* memfokuskan pada apa yang bisa dilakukan sistem untuk membantu *actor* dalam pekerjaan mereka. Hasil dari kegiatan *function* adalah daftar dari *function* (*function list*) yang lengkap, yang merinci *function – function* yang kompleks. *Function list* dibuat berdasarkan *use case description.*

Menurut Mathiassen *et al.* (2000, p138), terdapat empat tipe utama dari *function,* di mana masing – masing tipe mengekspresikan hubungan antara model dan konteks sistem. Keempat tipe tersebut ialah:

- *Update*

*Function* ini disebabkan oleh *event problem-domain* dan menghasilkan perubahan dalam *state* atau keadaan dari model tersebut

- *Signal*

*Function* ini disebabkan oleh perubahan keadaan atau *state* dari model yang dapat menghasilkan reaksi pada konteks.

- *Read*

*Function* ini disebabkan oleh kebutuhan informasi dalam pekerjaan *actor* dan mengakibatkan sistem menampilkan bagian yang berhubungan dengan informasi dalam model.

- *Compute*

*Function* ini disebabkan oleh kebutuhan informasi dalam pekerjaan *actor* dan berisi perhitungan yang melibatkan informasi yang disediakan oleh *actor* atau model, hasil dari *function* ini adalah tampilan dari hasil komputasi.

#### 2.6.8.4 *Interface*

*Interface* digunakan oleh aktor untuk berinteraksi dengan sistem. Menurut Mathiassen *et al.* (2000, p151), *interface* adalah suatu fasilitas untuk membuat suatu

sistem model dan fungsi-fungsi yang tersedia bagi aktor. *Interface* terbagi menjadi dua macam:

1. *User interfaces,* merupakan *interface* yang disediakan untuk pengguna. Menurut Mathiassen *et al.* (2000, p154-155), *user interfaces* terbagi menjadi empat, yaitu:

    a. *menu-selection pattern*, pola yang terdiri dari daftar pilihan yang mungkin dalam *interface* pengguna.

    b. *form fill-in,* merupakan pola klasik untuk *entry* data

    c. *command-languange pattern,* pola di mana pengguna memasukkan dan memulai *format* perintah sendiri

    d. *direct-manipulation pattern,* pola di mana pengguna memilih *object* dan melaksanakan *function* atas *object* dan melihat hasil dari interaksi mereka tersebut.

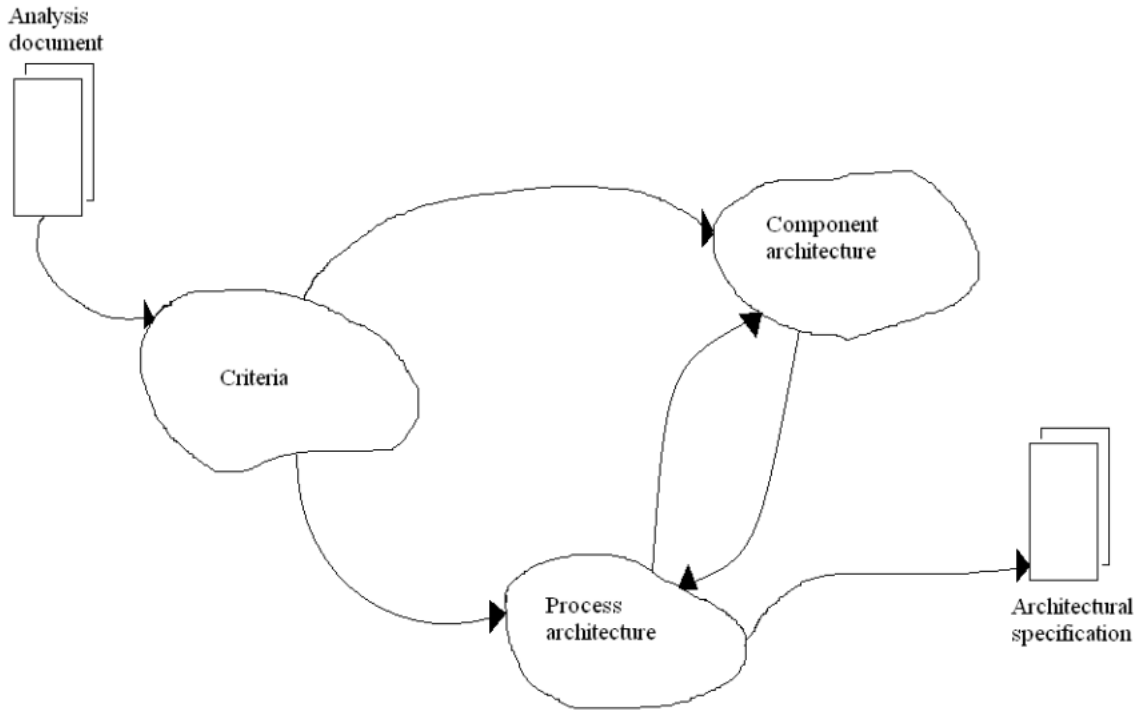
2. *System interface,* merupakan *interface* yang disediakan untuk sistem lain. Sistem lain dapat berupa: *external device* (misal: *sensor, switch*) dan sistem komputer yang kompleks sehingga dibutuhkan suatu protokol komunikasi.

### 2.6.9 *Architectural Design*

Mengacu pada Mathiassen *et al.* (2000), keberhasilan suatu sistem ditentukan dari kekuatan desain arsitekturalnya. Arsitektur membentuk sistem yang sesuai dengan sistem tersebut dengan memenuhi kriteria desain tertentu. Arsitektur ini juga berfungsi sebagai kerangka untuk pengembangan selanjutnya. Tujuan dari *architectural design* ini

adalah untuk membangun sistem komputerisasi. Pada akhirnya, arsitektur ini menghasilkan strktur untuk proses dan komponen sistem.

Menurut Mathiassen *et al.* (2000, p176), *architectural design* memiliki tiga subaktivitas, yaitu: *criteria, component,* dan *processes.*




Gambar 2.13 Aktivitas dalam *Architectural Design*
*Sumber : Mathiassen et al, 2000, p176*

### 2.6.9.1 *Criteria*

Menurut Mathiassen *et al.* ( 2000, p177), *criteria* merupakan suatu prioritas dari arsitektur. Tujuan aktivitas *criteria* adalah untuk menentukan prioritas dari sebuah perancangan. Adapun hasil dari *criteria* berupa kumpulan *criteria* yang telah diprioritaskan.

Tabel 2.3 Beberapa *Criteria* untuk Menentukan Kualitas *Software*

| ***Criterion*** | ***Measure of*** |
|---|---|
| *Usable* | Kemampuan adaptasi sistem terhadap konteks organisasi, berhubungan dengan kerja dan teknikal. |
| *Secure* | Suatu pencegahan melawan akses yang tidak terotorisasi terhadap data dan fasiltas yang ada. |
| *Efficient* | Eksploitasi yang ekonomis terhadap fasilitas *technical platform*. |
| *Correct* | Pemenuhan terhadap persyaratan – persyaratan. |
| *Reliable* | Pemenuhan terhadap ketelitian yang dibutuhkan dalam eksekusi fungsi. |
| *Maintainable* | Besarnya usaha atau biaya untuk menempatkan dan memperbaiki kecacatan sistem. |
| *Testable* | Besarnya usaha atau biaya untuk memastikan bahwa sistem yang dikembangkan menjalankan fungsi – fungsi yang dimaksudkan. |
| *Flexible* | Besarnya usaha atau biaya untuk memodifikasi sistem yang dikembangkan. |
| *Comprehensible* | Usaha yang dibutuhkan untuk mendapatkan pengertian yang logis terhadap sistem. |
| *Reusable* | Potensi penggunaan bagian – bagian sistem dalam sistem lain yang terhubung. |
| *Portable* | Besarnya usaha untuk memindahkan sistem ke *technical platform* lainnya. |
| *Interoperable* | Besarnya usaha untuk menggabungkan sistem ke sistem lain. |

*Sumber : Mathiassen et al, 2000, p178*

### 2.6.9.2 *Component Architecture*

Menurut Mathiassen *et al.* (2000, p190), *component Architecture* adalah struktur sistem yang terdiri dari komponen yang saling berhubungan. *Component* adalah kumpulan dari bagian – bagian program yang membentuk sistem dan memiliki tanggung jawab yang telah didefinisikan dengan jelas.

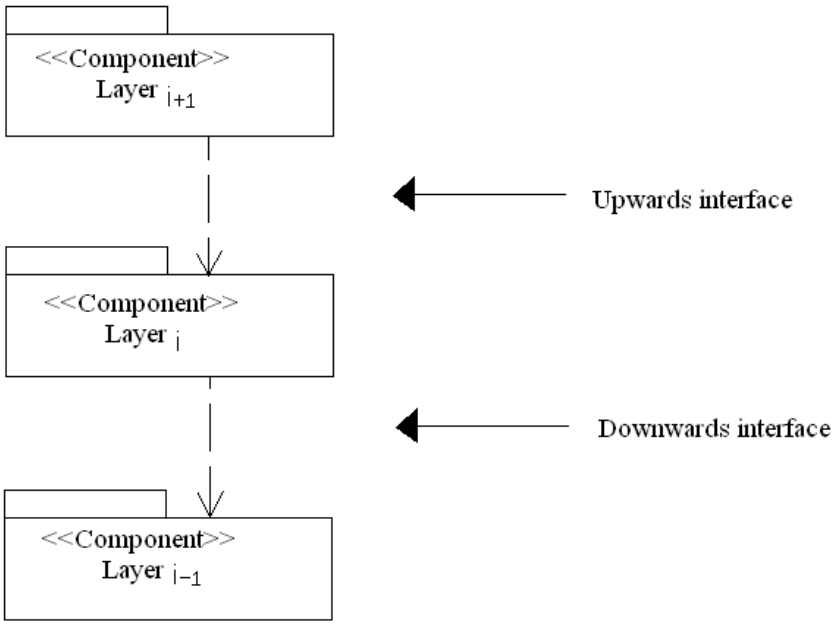
*Component architecture* yang baik membuat sistem lebih mudah untuk dimengerti, menyederhanakan desain dan mencerminkan kestabilan sistem. Adapun

tujuan dari *component architecture* adalah untuk menciptakan suatu struktur sistem yang *comprehensible* dan *flexible,* yang pada akhirnya, akan menghasilkan sebuah *class diagram* dengan spesifikasi dari komponen – komponen yang kompleks.

Menurut Mathiassen *et al.* (2000, p193-198), terdapat beberapa pola umum yang dapat digunakan untuk mendesain suatu *component architecture*:

***The Layered Architecture Pattern***

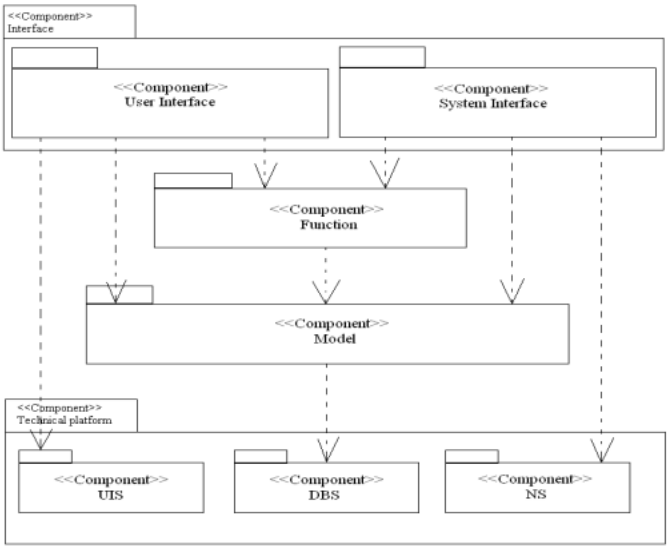
Arsitektur ini terdiri dari beberapa komponen yang didesain sebagai *layers*. Desain dari setiap komponen menggambarkan tanggung jawabnya masing – masing serta *interface*-nya ke arah atas maupun bawah. *Interface* ke bawah menggambarkan operasi mana yang dapat diakses oleh komponen dalam *layer* di bawahnya. *Interface* ke atas menggambarkan operasi yang tersedia untuk *layer* di atasnya.




Gambar 2.14 *Layered Architectural Pattern*
*Sumber : Mathiassen et al, 2000, p193*

***The Generic Architectural Pattern***

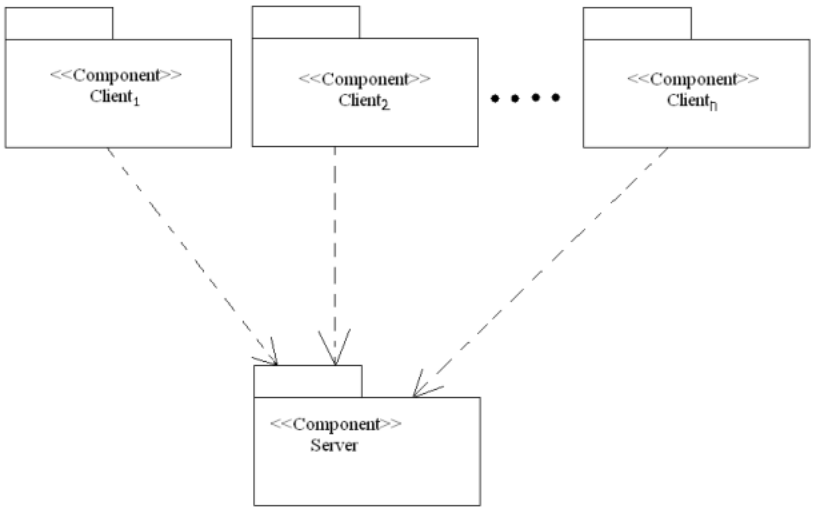
Model komponen mengandung model dari sistem *object,* yang dapat berupa *layer* yang paling bawah, kemudian diikuti dengan *system function layer,* dan yang paling atas merupakan *component interface. Layer interface* dapat dibagi menjadi dua bagian yang terpisah yaitu: *user interface* dan *system interface.*




Gambar 2.15 *The Generic Architectural Pattern*
*Sumber : Mathiassen et al, 2000, p196*

***The Client Server Architecture Pattern***

Pola arsitektur *client-server* sebenarnya dikembangkan untuk menangani distribusi dari sistem di antara beberapa *processors* yang tersebar secara geografis. Komponen dari arsitektur ini adalah sebuah *server* dan beberapa *client. Server* bertanggung jawab untuk menyediakan hal – hal yang umum bagi *client*-nya, seperti *database* atau sumber daya lain yang bisa digunakan bersama. *Server* seringkali menyediakan operasinya bagi *client* melalui suatu jaringan. *Client* bertanggung jawab untuk menyediakan *interface* lokal bagi para pengguna.




Gambar 2.16 *The Client Server Architecture Pattern*
*Sumber : Mathiassen et al, 2000, p197*

Ada beberapa jenis distribusi *client-server architecture* yang dapat dilihat pada tabel di bawah ini:

Tabel 2.4 Berbagai Bentuk Distribusi dalam *Client-Server Architecture*

| ***Client*** | ***Server*** | ***Architecture*** |
|---|---|---|
| U | U+F+M | *Distributed presentation* |
| U | U+F | *Local presentation* |
| U+F | F+M | *Distributed functionality* |
| U+F | M | *Centralized data* |
| U+F+M | M | *Distributed data* |

*Sumber: Mathiassen et al, 2000, p200*

### 2.6.9.3 *Process Architecture*

Menurut Mathiassen *et al.* (2000, p209), *process architecture* adalah struktur dari eksekusi sistem yang terdiri dari proses – proses yang saling tergantung. Tujuan dari *process architecture* ialah untuk mendefinisikan struktur fisik dari sebuah sistem, yang

pada akhirnya akan menghasilkan suatu *deployment diagram* dengan komponen program dan aktif *object*.

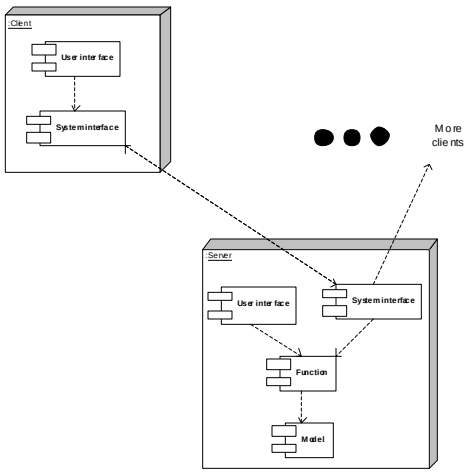
*Process architecture* membawa kita lebih dekat ke tingkat fisikal sistem yang berfokus pada distribusi dan eksekusi, dan pekerjaan dengan proses dan *object*. Aktivitas *process architecture* itu sendiri terbagi menjadi dua tingkat abstraksi. Tingakat yang pertama ialah tingkat keseluruhan (*overall level*) di mana kita mendefinisikan distribusi dari komponen progra yang tersedia di *system processor*. Tingkat yang kedua berhubungan dengan proses yang menstruktur kolaborasi di antara *object* selama masa ekesekusi.

Menurut Mathiassen *et al.* (2000, p212), *process architecture* memiiliki beberapa subaktivitas, yaitu: *distributed program components, explorer distribution pattern, identify shared reources, select coordination mechanism,* dan *explore coordination pattern.*

Menurut Mathiassen *et al.* (2000, p215-218), ada tiga pola distribusi pada aktivitas desain *process architecture*:

1. *Centralized pattern*

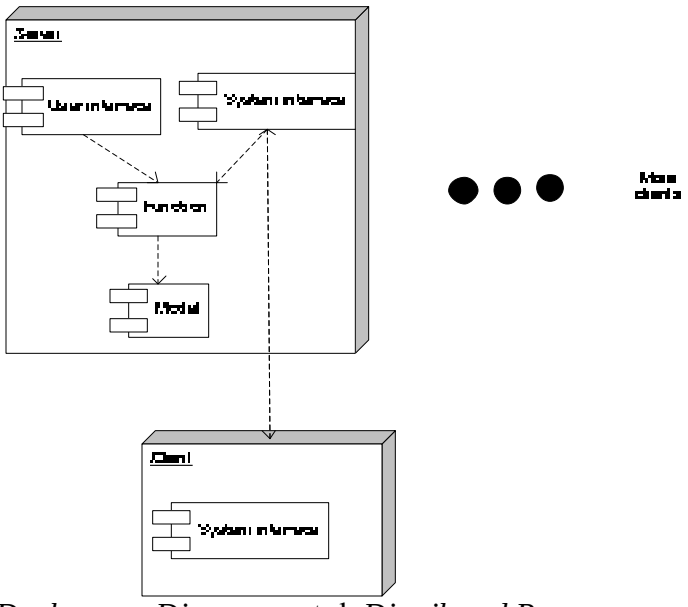
Pada pola ini semua data ditempatkan pada *server* dan *client* hanya menangani *user interface* saja. Kesluruhan model dan semua fungsi bergantung pada *server*,dan *client* hanya berperan seperti terminal.




Gambar 2.17 *Deployment Diagram* untuk *Centralized Pattern*
*Sumber : Mathiassen et al, 2000, p216*

2. *Distributed pattern*

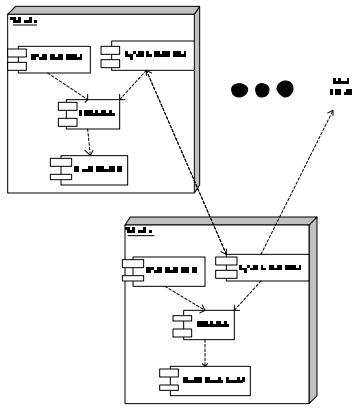
Pola ini merupakan kebalikan dari *centralized pattern*. Pada pola ini, semua didistribusi kepada *client* dan *server* hanya diperlukan untuk melakukan *update* model di antara *clients*.



Gambar 2.18 *Deployment Diagram* untuk *Distributed Pattern*
*Sumber : Mathiassen et al, 2000, p217*

3. *Desentralized pattern*

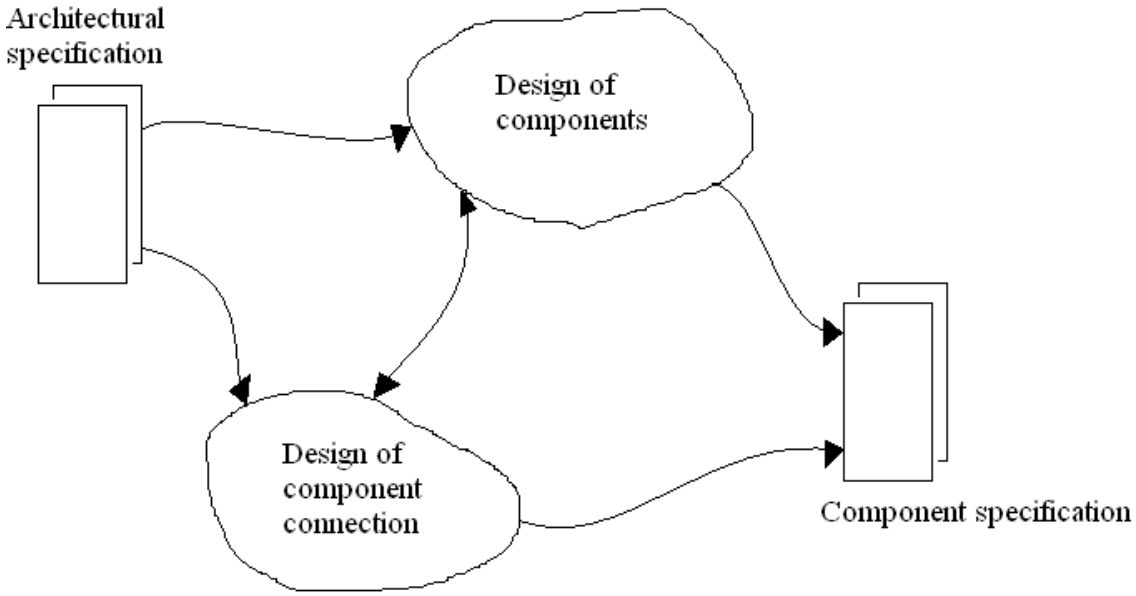
Pola ini dapat dikatakan merupakan gabungan dari kedua pola sebelumnya. Pada pola ini, *client* mengimplementasikan model yang lokal, sedangkan *server* akan memakai model umum (*common*).



Gambar 2.19 *Deployment Diagram* untuk *Desentralized Pattern*
*Sumber : Mathiassen et al, 2000, p219*

### 2.6.10 *Component Design*

Menurut Mathiassen *et al.* (2000, p231-233), *component design* bertujuan untuk menentukan implementasi kebutuhan dalam rangka kerangka arsitektural. Kegiatan desain komponen bermula dari spesifikasi arsitektural dan kebutuhan sistem, sedangkan hasil dari kegiatan ini adalah spesifikasi dari komponen yang saling berhubungan.




Gambar 2.20 Subaktivitas pada *Component Design*
*Sumber : Mathiassen et al, 2000, p232*

### 2.6.10.1 *Model Component*

Menurut Mathiassen *et al.* (2000, p236), *model component* adalah bagian dari sistem yang mengimplementasi model *problem domain*. Tujuan dari komponen model adalah untuk mengirimkan data sekarang dan *historic* ke *function*, i*nterface* dan pengguna dan sistem yang lain. Konsep utama dalam desain *model component* adalah struktur.

Hasil dari kegiatan *model component* adalah revisi dari *class diagram* dari kegiatan analisis. Kegiatan revisi biasanya terdiri dari kegiatan menambahkan kelas, *attribute*, dan struktur baru yang mewakili *event*.

Menurut Mathiassen *et al.* (2000, p239), revisi *class diagram* dapat dilakukan dengan memperhatikan *private events* dan *common events*. *Private events* adalah *event* yang melibatkan hanya satu *object domain*.

Menurut Mathiassen *et al.* (2000, p239-240), langkah – langkah atau panduan dalam mempresentasikan *privates events:*

*Event-event* yang hanya terjadi pada *sequence dan selection*:

- Representasikan *event-event ini* sebagai *state attribute* pada *class* yang digambarkan oleh *statechart diagram*. Setiap kali ada salah satu dari *events* yang terlibat timbul, maka sistem akan menugaskan nilai yang baru kepada *state attribute.*
- Integrasikan *attribute*dari *event* yang terlibat ke dalam class.

*Event-event* yang terjadi 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 dari *class.*
- Integrasikan *attribute event* ke dalam *class* yang baru. Jika suatu *event* adalah *common* / 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.

Menurut Mathiassen *et al.* (2000, p241), langkah – langkah atau panduan dalam mempresentasikan *common events:*

*Common Event*:

- Jika *event* yang terlibat dalam *statechart diagram* dalam cara yang berbeda, representasikan *event* tersebut dalam hubungannya ke class yang menawarkan representasikan paling sederhana.
- Jika *event* yang terlibat dalam statechart diagram dalam cara yang sama, pertimbangkan alternatif representasi yang mungkin antara satu sama lain.

Untuk menyederhanakan *class diagram* yang telah direvisi dari hasil tahapan sebelumnya, dilakukan restrukturisasi *class* baik melalui *generalization, association*, maupun *embedded iteration.*

#### 2.6.10.2 *Function Component*

Menurut Mathiassen *et al.* (2000, p252), *function component* adalah bagian dari sistem yang mengimplementasikan kebutuhan fungsional. Tujuan dari *function component* adalah untuk memberikan akses bagi *user interface* dan komponen sistem lainnya ke model, oleh karena itu *function component* adalah penghubung antara model dan *usage.* Sedangkan tujuan dari *function component design* adalah menentukan implementasi *functions*. Hasil dari kegiatan ini adalah *class diagram* dengan *operations* dan spesifikasi dari *operations* yang kompleks.

*Function* didesain dan diimplementasikan dengan menggunakan operasi dari class sistem. Operasi adalah proses yang dispesifikasikan dalam sebuah *class* dan dijalankan melalui *object* dari *class* tersebut .

Sub kegiatan dalam *component function* akan menghasilkan kumpulan operasi yang dapat mengimplementasikan fungsi sistem seperti yang ditentukan dalam analisis *problem domain* dan *function list.*

Adapun sub kegiatan dalam *component function*:

1. Merancang *function* sebagai *operation.*

   Yaitu mengidentifikasi tipe utama dari *functions* tersebut. Ada empat tipe *functions* (Mathiassen *et al,* 2000, p225-260), yaitu: *Update, Read, Compute,* dan *Signal.*

2. Menelusuri pola yang dapat membantu dalam implementasi *function* sebagai *operation.*

Empat pola menurut Mathiassen *et al.* (2000, p260-264) adalah:

a. *Model Class Placement*

Pola ini menempatkan *operation* dalam *model component class* dan berguna ketika sebuah *operation* mengakses hanya 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.*

b. *Function Class Placement*

Pola ini digunakan ketika tanggung jawab *operation* tidak dapat dengan jelas ditempatkan dalam *model class*. Sebaliknya satu atau lebih *functional-component class* dapat digambarkan dengan menempatkan *operation* yang merealisasikan *function.*

c. *Strategy*

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

d. *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 *performance*-nya tergantung dari *state* pada *model component.*

3. Spesifikasikan operasi yang kompleks.

   Menurut Mathiassen *et al.* (2000, p271) ada tiga cara untuk melakukannya, yaitu *operation spesification, sequence diagram,* dan *statechart diagram.*

#### 2.6.10.3 *Connecting Component*

Menurut Mathiassen *et al.* (2000, p271), *connecting component* digunakan untuk 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 flexible* dan *comprehensible*. Pada *Connecting component* ada dua konsep yaitu:

- *Coupling* adalah suatu ukuran yang digunakan untuk menentukan bagaimana dekatnya hubungan antara dua *class* atau *component.*
- *Cohesion* merupakan ukuran seberapa kuatnya keterikatan dari suatu *class* atau *component.*

Prinsipnya adalah:"*highly cohesive classes and loosely coupled components*".

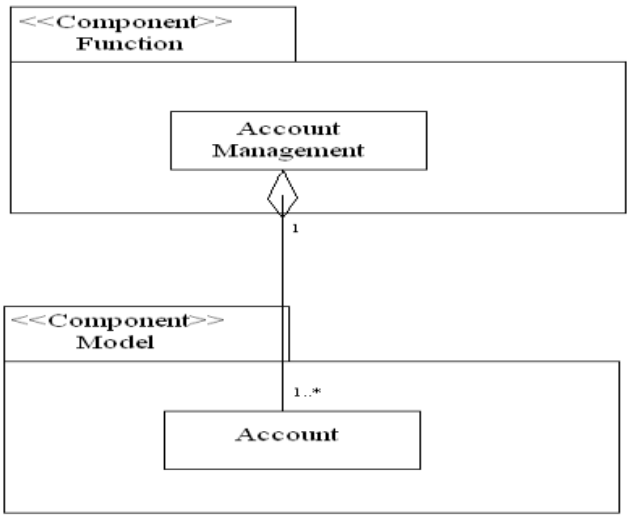
Adapun aktivitas yang terkait dalam mendesain koneksi di antara komponen adalah:

a. Menghubungkan *class – class*
b. Mengeksplorasi polanya
c. Melakukan evaluasi terhadap koneksi yang ada

Hasil dari aktivitas connecting components ini adalah *class diagram* yang di mana *dependencies*-nya berubah menjadi *connections*.

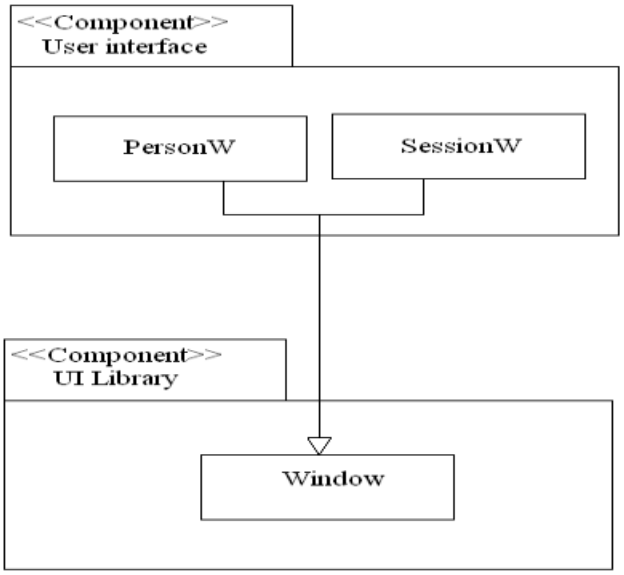
Menurut Mathiassen *et al.* (2000, p275-277) adalah:

- *Class aggregation*, yaitu mengagregasikan *class-class* dari *component* lain. Koneksi ini beguna ketika *class definition* sudah ada di dalam *component* lain. Umumnya *coupling*-nya rendah, namun sulit mencapai *cohesive.*



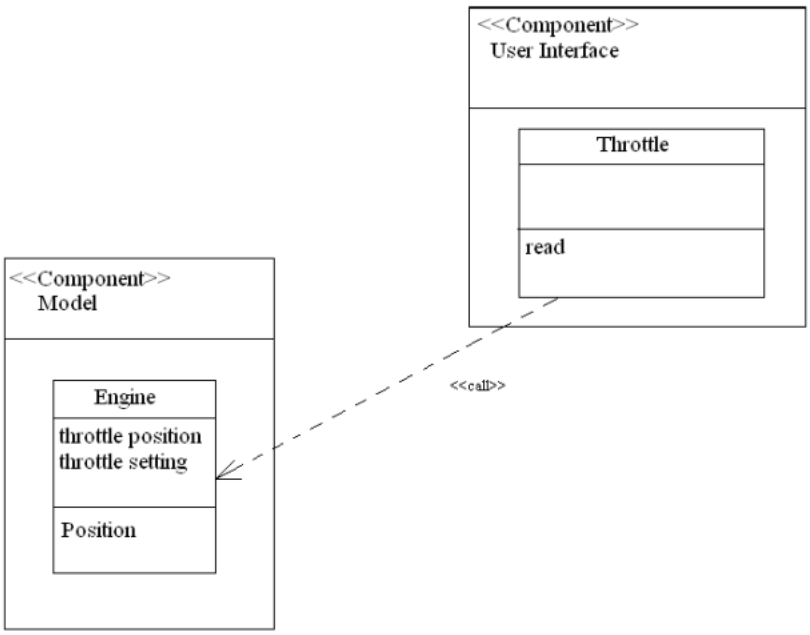

Gambar 2.21 *Connection* oleh *Class Aggregation*
*Sumber : Mathiassen et al, 2000, p275*

- *Class specialization*, yaitu menspesialisasikan public *class* dari *component* lain.




Gambar 2.22 *Connection* oleh *Class Specialization*
*Sumber : Mathiassen et al, 2000, p276*

- *Operation call*, yaitu memanggil *public operations* di dalam *object-object* dari *component* lain. Umumnya *coupling*-nya rendah dan *cohesion*-nya tinggi.




Gambar 2.23 *Connection* oleh *Operation Call*
*Sumber : Mathiassen et al, 2000, p277*

## 2.7 Teori Matriks

Menurut Gelinas dan Dull (2008, p285), matriks pengendalian adalah alat yang didesain untuk membantu dalam mengevaluasi pengendalian yang efektif dalam proses bisnis tertentu dengan menyesuaikan tujuan pengendalian dengan rencana pengendalian yang relevan. PCAOB *Auditing Standard* Nomor 2 menyebutnya sebagai "efektivitas dari desain pengendalian". Menilai efektivitas dari desain pengendalian sesuai dengan SOX Nomor 404. Ketika managemen dan auditor independen melakukan penilaian ini, biasanya mereka menggunakan matriks pengendalian.

Menurut Gelinas dan Dull (2008, p293), langkah–langkah untuk membuat matriks pengendalian adalah:

1. Spesifikasikan tujuan pengendalian: *review flowchart* sistem dan deskripsi naratif yang terkait agar menjadi lebih familiar dengan sistem yang diperiksa. Identifikasi proses bisnis; sumberdaya relevan utama; input; penyimpanan, jika ada untuk data input; dan data master di-*update*.
   a. Identifikasi tujuan pengendalian proses operasi:
      - *Effectiveness goals* (mungkin lebih dari satu).
      - *Efficiency goals* (biasanya orang dan komputer).
      - *Secutity goals* (termasuk semua data yang mempengaruhi dan aset berwujud).
   b. Identifikasi tujuan proses informasi:
      - *Input goals* (validitas, kelengkapan dan akurasi).
      - *Update goals* (kelengkapan dan akurasi), jika prosesnya periodik
2. Rekomendasikan rencana pengendalian: buat daftar semua rencana pengendalian yang direkomendasikan yang sesuai dengan proses yang dianalisa. Daftar ini harus terdiri baik dari rencana yang terkait dengan operasi dan yang terkait dengan metode pemrosesan operasi.
   a. Pada *present control* yang ada pada *flowvhart* sistem, tambahkan di depannya keterangan seperti P-1, P-2 dan seterusnya. Mulai dari kolom paling atas sebelah kiri dari *flowchart*, terus ikuti urutan proses *flowchart*.
   b. Evaluasi *present control plan* dengan menempatkan nomor dan nama dari rencana matriks pengendalian dan memberikan penjelasan di bagian bawah matriksnya.
   c. Identifikasi dan evaluasi rencana pengendalian yang tidak ada.

- Periksa matriks pengendalian untuk menentukan jika ada tujuan pengendalian yang tidak memiliki rencana pengendalian. Jika ada, buat rencana pengendalian yang didesain untuk meminimalkan risiko yang terkait (tujuan pengendalian). Kemudian, berikan penjelasan di bagian bawah matriks. Ulangi prosedur ini sampai seluruh tujuan pengendalian pada matriks telah memiliki satu atau lebih rencana pengendalian.
- Analisa *flowchart* sistem untuk mengetahui risiko yang akan datang sehingga kita dapat menambah perencanaan pengendalian untuk mencegah atau memperkuat pengendalian yang ada. Perbaikan dari matriks pengendalian menggunakan prosedur yang sama dideskripsikan untuk *present or missing control plans*.