BAB 2

LANDASAN TEORI

2.1. Kualitas

2.1.1. Definisi Kualitas

Kualitas dapat menjadi konsep yang berbeda bagi beberapa orang, pengertian kualitas terus berevolusi seiring dengan pertumbuhan dan kedewasaan profesi yang berhubungan dengan kualitas. Tidak ada satu pun konsultan maupun pelaku bisnis yang setuju pada satu pengertian kualitas yang universal. Sebuah penelitian yang menanyakan tentang definisi kualitas pada manajer 86 perusahaan di bagian timur Amerika Serikat menghasilkan beberapa jawaban yang berbeda, diantaranya (Evans et al., 2007, p12):

1. Kesempurnaan
2. Konsistensi
3. Pengurangan limbah
4. Kecepatan pengiriman
5. Ketaatan pada peraturan dan prosedur
6. Penyediaan produk yang baik dan bermanfaat
7. Melakukan hal yang benar sejak awal
8. Memuaskan pelanggan
9. Pelayanan pelanggan secara total dan memuaskan

Definisi tersebut dapat berarti bahwa pengertian kualitas dari berbagai paradigma dapat membantu kita dalam memahami peran kualitas di berbagai bagian dari sebuah organisasi.

Menurut American Society for Quality Control yang mengatakan, Quality is the totality of features and characteristics of a product or service that bear on its ability to satisty stated of implied needs

Berikut ini merupakan pendapat dari para ahli mengenai pengertian kualitas diantaranya, yaitu (Ariani, 2003):

• Menurut Juran (1962)

"Kualitas adalah kesesuaian dengan tujuan atau manfaatnya."

• Menurut Crosby (1979)

"Kualitas adalah kesesuaian dengan kebutuhan yang meliputi *availabity*, *delivery*, *reliability, maintainability*, dan *cost effectiveness.*"

• Menurut Deming (1982)

"Kualitas harus bertujuan memenuhi kebutuhan pelanggan sekarang dan di masa mendatang."

• Menurut Feigenbaum (1991)

"Kualitas merupakan keseluruhan karakteristik produk dan jasa yang meliputi *marketing, engineering, manufacture*, dan *maintenance*, dalam mana produk dan jasa tersebut dalam pemakaiannya akan sesuai dengan kebutuhan dan harapan pelanggan."

• Menurut Scherkenbach (1991)

"Kualitas ditentukan oleh pelanggan; pelanggan menginginkan produk dan jasa yang sesuai dengan kebutuhan dan harapannya pada suatu tingkat harga tertentu yang menunjukkan nilai produk tersebut."

• Menurut Elliot (1993)

“Kualitas adalah sesuatu yang berbeda untuk orang yang berbeda dan tergantung pada waktu dan tempat, atau dikatakan sesuai dengan tujuan.”

• Menurut Goetch dan Davis (1995)

“Kualitas adalah suatu kondisi dinamis yang berkaitan dengan produk, pelayanan, orang, proses, dan lingkungan yang memenuhi atau melebihi apa yang diharapkan.”

• Perbendaharaan istilah *ISO* 8402 dan dari Standar Nasional Indonesia (SNI 19-8402-1991)

“Kualitas adalah keseluruhan ciri dan karakteristik produk atau jasa yang kemampuannya dapat memuaskan kebutuhan, baik yang dinyatakan secara tegas maupun tersamar. Istilah kebutuhan diartikan sebagai spesifikasi yang tercantum dalam kontrak maupun kriteria-kriteria yang harus didefinisikan terlebih dahulu.”

2.1.2. Pengertian Pengendalian Kualitas

Menurut Sofjan Assauri dalam buku Manajemen Produksi dan Operasi (2004 :p210) dikatakan bahwa ”Pengendalian kualitas adalah kegiatan memastikan apakah kebijakan dalam hal kualitas (standar) dapat tercermin dalam hasil akhir, atau dengan kata lain usaha untuk mempertahankan mutu atau kualitas dari barang-barang yang dihasilkan agar sesuai dengan spesifikasi produk yang telah ditetapkan berdasarkan kebijakan pimpinan”. Dalam mewujudkan pelaksanaan dari pengendalian kualitas, kegiatan ini dilakukan oleh operator dan manajemen dari departemen yang bersangkutan dengan melakukan pengukuran pencapaian standar yang telah ditetapkan sebelumnya.

2.2 *Six sigma*

2.2.1 Pengertian *Six sigma*

Menurut Evans dan Lindsay (2007) Six Sigma didefinisikan sebagai metode peningkatan proses bisnis yang bertujuan untuk menemukan dan mengurangi faktor-faktor penyebab kecacatan dan kesalahan, mengurangi waktu siklus dan biaya operasi, meningkatkan produktivitas, memenuhi kebutuhan pelanggan dengan lebih baik, mencapai tingkat pendayagunaan aset yang lebih tinggi, serta mendapatkan hasil atas investasi yang lebih baik dari segi produksi maupun pelayanan

Tujuan Six Sigma adalah untuk meningkatkan kinerja bisnis dengan mengurangi berbagai variasi proses yang merugikan, mereduksi kegagalan-kegagalan produk/ proses, menekan cacat-cacatproduk, meningkatkan keuntungan, mendongkrak moral karyawan, dan meningkatkan kualitas produk pada tingkat yang maksimal.

2.2.2 Metodologi DMAIC

DMAIC (Define Measure Analyze Improve Control) merupakan sebuah komponen dasar dari metodologi Six Sigma, yang digunakan untuk meningkatkan kinerja suatu proses dengan mengeliminasi defect. DMAIC dikembangkan oleh Edwards Deming dan berguna untuk memperbaiki sebuah proses bisnis untuk mengurangi cacat produksi. Adapun fase-fase dari DMAIC adalah sebagai berikut (Breyfogle., 2003, p45):

Gambar 2.1 Siklus *DMAIC*

Berikut ini adalah penjelasan singkat dari metode *DMAIC*:

2.2.2.1. Fase *Define*

Hal-hal penting yang dapat didefinisikan pada fase *Define* yaitu suara pelanggan (*Voice of Costumer*) yang ditransformasikan kedalam karakteristik penting kualitas, ruang lingkup proyek, prioritas sebab akibat serta perencanaan proyek. Sebuah permasalahan harus bersumber dari data yang ada, dapat diukur, dan lepas dari asumsi tentang penyebab atau penyelesaian masalah yang diperkirakan. Oleh karena itu, permasalahan yang akan ditanggulangi harus spesifik dan tujuannya dapat dicapai.

Berikut ini adalah *tools* yang dapat dipakai pada fase *Define* diantaranya adalah:

a. *Brainstorming*

Brainstorming merupakan alat yang dapat digunakan untuk menghasilkan ide, *brainstorming* juga dapat berguna untuk merangsang ide kreatifitas dalam berpikir.

b. Diagram *SIPOC* (*Supplier, Input, Process, Output, Customer)*

Dalam manajemen dan perbaikan proses, diagram SIPOC merupakan salah satu teknik yang paling berguna dan juga paling sering digunakan.

Diagram *SIPOC* merupakan singkatan dari *Supplier – Input – Process – Output – Customer* yang digunakan untuk menampilkan aliran kerja secara luas. Berikut ini adalah defnisi dari *SIPOC:*

a. *Supplier* adalah orang atau sekelompok orang yang memberikan informasi kunci, material atau sumber daya lain kepada proses. *Supplier* dapat juga merupakan proses sebelum proses yang menjadi fokus.

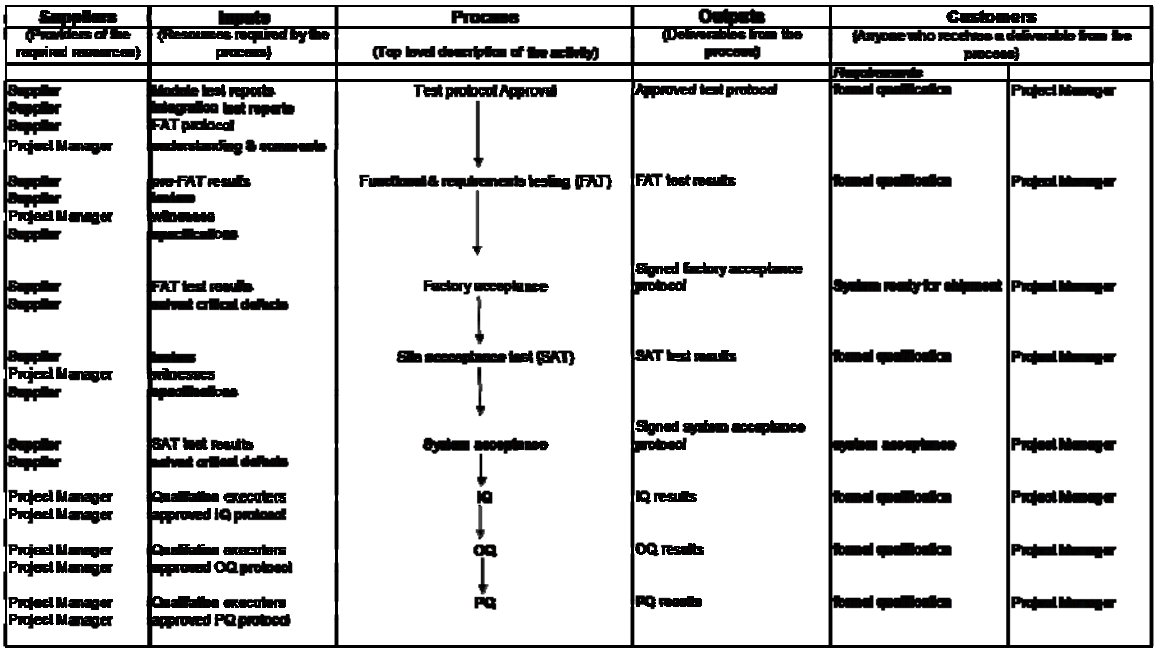
b. *Input* adalah segala sesuatu yang diberikan pemasok ke dalam proses bisnis perusahaan.

c. *Process* adalah sekumpulan langkah yang mengubah input sehingga memberikan nilai tambah pada input.

d. *Output* adalah produk atau proses *final*, bisa berupa barang ataupun jasa yang dihasilkan lewat suatu proses.

e. *Customer* menurut Mulyadi (2001:224) adalah “siapa saja yang menggunakan keluaran pekerjaan seseorang atau suatu tim.” Dalam hal ini, customer dapat bersifat intern maupun ekstern dari sudut organisasi. Customer tidak dapat disamakan dengan pelanggan. Pelanggan mempunyai pengertian sebagai pembeli berulang kali (*repeat buyer*). Sedangkan customer mencakup *repeat buyer* dan *on time buyer.*

.



Gambar 2.2 Contoh Diagram *SIPOC*

Sumber: www. leanvalidation.eu

2.2.2.2. Fase *Measure*

Fase kedua dilakukan pada saat memulai mengumpulkan data tentang kinerja saat ini. Pada saat fase *measure* berlangsung, pengolahan data harus sesuai dengan tipe data yang dimiliki, dengan demikian, pengukuran yang valid akan menjamin akurasi dan konsistensi, kecukupan data untuk analisis, dan sebuah gambaran analisis awal untuk mengarahkan proyek. Inti dari fase measure ini yaitu mengembangkan perencanaan pengumpulan data, mengidentifikasi variabel inti masukan proses, menampilkan variasi dengan *tools* yang tepat. Berikut ini pengukuran data yang dibedakan menjadi 2 yaitu :

- **Data Atribut**

  Merupakan data kualitatif yang dapat dihitung untuk pencatatan dan analisis. Contoh dari data atribut karakteristik kualitas adalah ketiadaan label pada kemasan produk,kesalahan

proses administrasi buku tabungan nasabah, banyaknya jenis cacat pada produk. Data atribut biasanya diperoleh dalam bentuk unit-unit nonkonformasi atau ketidaksesuaian dengan spesifikasi atribut yang telah ditetapkan.

- **Data Variabel**

  Adalah data kuantitatif yang diukur untuk keperluan analisis. Contoh dari data variable karakteristik kualitas adalah :diameter pipa, berat semen dalam kantong, banyaknya kertas tiap rim,dll. Ukuran-ukuran seperti volume, berat, panjang, lebar, tinggi, diameter, adalah merupakan data variable.

Pada tingkatan proyek *six sigma,* indikator kualitas produk / jasa biasanya berfokus pada output dari proses manufaktur atau jasa. Salah satu indicator kualitas manufaktur yang umum ditemui adalah jumlah *Defect Per Unit* (DPU). Berdasarkan nilai dari DPU kemudian dapat ditentukan nilai dari *Defect Per Million Opportunities* (DPMO) untuk dapat menentukan tingkatan sigma dari proses yang ada saat ini. Penentuan nilai sigma dapat dilakukan melalui rumus-rumus berikut :

$$\textit{Defect per Unit}\ (\text{DPU}) = \frac{\textit{Total Defect}}{\textit{Jumlah Output}}$$

$$\textit{Defect Per Million Opportunity}\ (\text{DPMO}) = \frac{DPU \times 1{,}000{,}000}{\textit{TotalKriteria CTQ}}$$

Tabel 2.1 Konversi Sigma

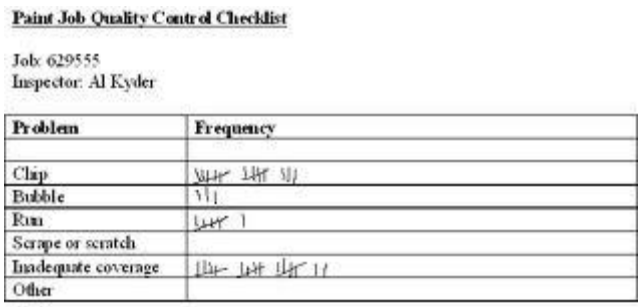
THE SIGMA TABLE					
Sigma	DPMO	Yield	Sigma	DPMO	Yield
6	3.4	99.99966%	3	66807	93.3%
5.9	5.4	99.99946%	2.9	80757	91.90%
5.8	8.5	99.99915%	2.8	96801	90.3%
5.7	13	99.99866%	2.7	115070	88.5%
5.6	21	99.9979%	2.6	135666	86.4%
5.5	32	99.9968%	2.5	158655	84.1%
5.4	48	99.9952%	2.4	184060	81.6%
5.3	72	99.9928%	2.3	211855	78.8%
5.2	108	99.9892%	2.2	241964	75.8%
5.1	159	99.984%	2.1	274253	72.6%
5	233	99.977%	2	308538	69.1%
4.9	337	99.966%	1.9	344578	65.5%
4.8	483	99.952%	1.8	382089	61.8%
4.7	687	99.931%	1.7	420740	57.9%
4.6	968	99.90%	1.6	460172	54.0%
4.5	1350	99.87%	1.5	500000	50.0%
4.4	1866	99.81%	1.4	539828	46.0%
4.3	2555	99.74%	1.3	579260	42.1%
4.2	3467	99.65%	1.2	617911	38.2%
4.1	4661	99.53%	1.1	655422	34.5%
4	6210	99.38%	1	691462	30.9%
3.9	8198	99.18%	0.9	725747	27.4%
3.8	10724	98.90%	0.8	758036	24.2%
3.7	13903	98.60%	0.7	788145	21.2%
3.6	17864	98.20%	0.6	815940	18.4%
3.5	22750	97.70%	0.5	841345	15.9%
3.4	28716	97.10%	0.4	864334	13.6%
3.3	35930	96.40%	0.3	884930	11.5%
3.2	44565	95.50%	0.2	903199	9.7%
3.1	54799	94.50%	0.1	919243	8.1%

Sumber : http://thequalityweb.commeasure4.html

Berikut ini adalah *tools* yang digunakan dalam tahapan *Measure*:

1. Lembar *Check Sheet*

Lembar *Check Sheet* merupakan sejenis formulir pengumpulan data khusus yang hasilnya dapat diinterpretasikan pada formuir tersebut secara langsung tanpa membutuhkan pemrosesan lebih lanjut.



Paint Job Quality Control Checklist

Job: 629555
Inspector: Al Kyder

Problem	Frequency
Clap	
Bubble	
Run	
Scrape or scratch	
Inadequate coverage	
Other	

Gambar 2.3 Contoh *Check Sheet*

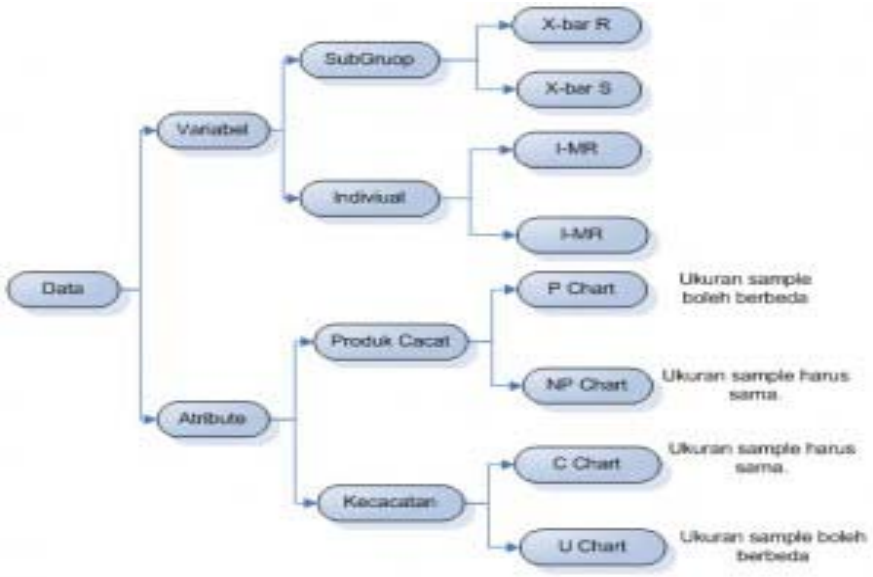
Sumber : http://syque.com/quality_tools/toolbook/Check/vary.htm

2. *Control Chart* (Peta Kendali)

C*ontrol chart* atau peta kendali adalah grafik yang digunakan untuk mengkaji perubahan proses dari waktu ke waktu. Merupakan salah satu alat atau *tool* dalam pengendalian proses secara statististik yang sering kita kenal dengan SPC (*Statistical Process Control*), ada juga yang menyebutnya dengan *Seven Tools*. Pembuatan *control chart* dalam SPC bertujuan untuk mengidentifikasi setiap kondisi didalam proses yang tidak terkendali secara statistik (*out of control*).

Dalam proses pembuatan *control chart* sangat penting memperhatikan jenis data yang kita miliki untuk menentukan jenis *control chart* yang tetap, sehingga dapat memberikan

informasi yang tetap terhadap kinerja proses. Kesalahan pemilihan jenis *control chart* dapat berakibat fatal, karena tidak ada informasi yang bisa tarik dari data yang sudah dikumpulkan bahkan dapat memberikan gambaran yang salah terhadap kinerja proses. *Control chart* dapat diklasifikasi seperti berikut:



Sumber: http://purdianta.com/?p=63

Gambar 2.4 Klasifikasi *Control Chart*

Ciri khas dari *control chart* baik untuk dapat *variabel* maupun *atribute* selalu di batas oleh batas kendali atas (*Upper Control Limit*) dan batas kendali bawah (*Lower Control Limit*). Peta kendali X-bar R sebenarnya lebih baik digunakan dari pada X-bar S karena dalam menggambarkan variasi yang terjadi didalam sample dari setiap sub group, sedangkan dalam X-bar R hanya menunjukan rentang nilai sample dalam masing-masing sub grup.

P *Chart* digunakan untuk pengendalian proporsi produksi cacat, ukuran sample yang dalam pembuatan P *chart* dapat berbeda antara suatu sub group dengan sub group yang lainnya.

Sedikit berbeda dengan NP chart, digunakan untuk memonitor jumlah produk cacat dan ukuran sample sub group datanya harus sama.

Peta kendali – p

Perbandingan antara banyaknya cacat dengan semua pengamatan, yaitu setiap produk yang diklasifikasikan sebagai “diterima” atau “ditolak” (yang diperhatikan banyaknya produk cacat).

Langkah-langkah pembuatan peta kendali p:

a. Tentukan ukuran contoh/subgrup yang cukup besar (n > 30)

b. Kumpulkan banyaknya subgrup (k) sedikitnya 20–25 sub-grup

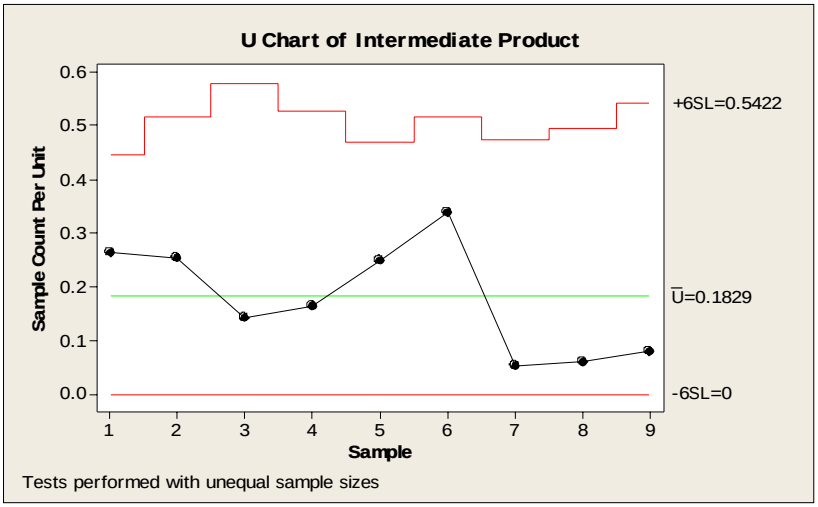
c. Hitung untuk setiap subgrup nilai proporsi unit yang cacat, yaitu:

p = jumlah unit cacat/ukuran subgrup

d. Hitung nilai rata-rata dari p, yaitu p dapat dihitung dengan:

p = total cacat/total inspeksi.

Jika yang ingin kita kembalikan kecacatan dari suatu produk, maka control chart yang dapat digunakan C *chart* dan U *chart.* Untuk pengendalian terhadap jenis cacat maka harus menggunakan C *chart,* sedangkan U *Chart* digunakan untuk pengendalian terhadap jumlah cacat per unit.



Gambar 2.5 Contoh *Control Chart*

Berikut ini adalah rumus-rumus yang digunakan dalam membuat berbagai jenis *control chart.*

a. Peta Kontrol X-Bar

Center Line (CL) = X-Double Bar

Upper Center Line (UCL) = X-Double Bar + (2)(A_2 R-Bar)

Lower Center Line (LCL) = X-Double Bar - (2)(A_2 R-Bar)

b. Peta Kontrol R-Bar

Center Line (CL) = R- Bar

Upper Center Line (UCL) = (2)(D_4 R-Bar)

Lower Center Line (LCL) = (2)(D_3 R-Bar)

c. Peta Kontrol X

Center Line (CL) = X- Bar

Upper Center Line (UCL) = X-Bar + (2)(MR-Bar/ D_2)

Lower Center Line (LCL) = X-Bar - (2)(MR-Bar/ D_2)

d. Peta Kontrol R

Center Line (CL) = MR- Bar

Upper Center Line (UCL= (2)(D_4 MR-Bar)

Lower Center Line (LCL) = (2)(D_3 MR-Bar)

e. Peta Kontrol P

Center Line (CL) = p-bar

Upper Center Line (UCL) = p-bar + $6S_p$

Lower Center Line (LCL) = p-bar - $6S_p$

f. Peta Kontrol NP

Center Line (CL) = np-bar

Upper Center Line (UCL= np-bar + $6S_{np}$

Lower Center Line (LCL) =np-bar - $6S_{np}$

g. Peta Kontrol C

Center Line (CL) = c-bar

Upper Center Line (UCL) = c-bar + $6S_c$

Lower Center Line (LCL) = c-bar - $6S_c$

h. Peta Kontrol U

Center Line (CL) = u-bar

Upper Center Line (UCL= u-bar + $6S_u$

Lower Center Line (LCL) =u-bar - $6S_u$

2.2.2.3. Fase *Analyze*

Pada fase *analyze*, fokus terhadap permasalahan sudah harus jelas. Dengan demikian, pada fase ini sudah dapat dilakukan analisis perbaikan dengan melihat data yang telah diolah. Sehingga fase analisis ini dapat digunakan untuk mencari penyebab munculnya masalah dan kemungkinan perbaikan yang akan diambil.

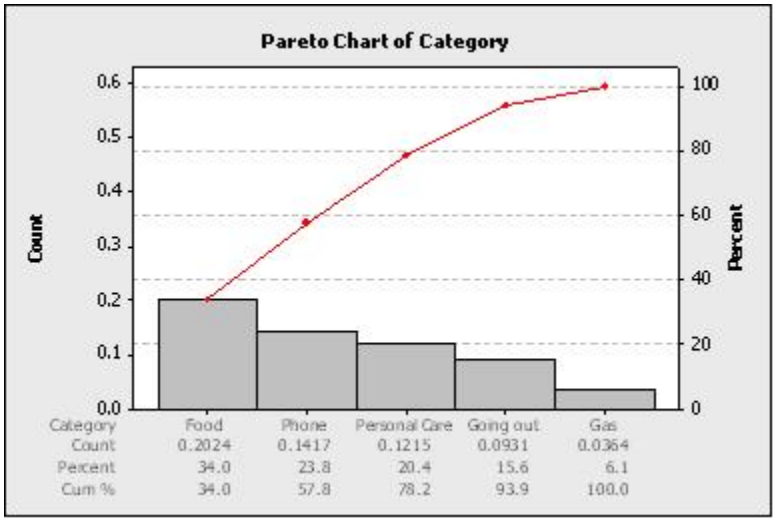
Berikut ini adalah *tools* yang dapat digunakan dalam fase *Analyze*:

a. Diagram Pareto

Dale H Besterfield (1994, p15) Vilfredo Pareto (1848-1923), seorang pakar ekonomi dalam pembelajarannya mengenai distribusi kekayaan di eropa, ia menemukan bahwa terdapat

sedikit orang yang memiliki banyak uang, dan banyak orang dengan sedikit uang. Perbedaan distribusi kekayaan ini menjadi bagian yang turun temurun dalam teori ekonomi.

Gerald Smith (1995, p5) diagram pareto adalah sejumlah kejadian yang spesifik yang digambarkan dalam diagram batang, bar terbesar menggambarkan permasalahan utama atau yang terbesar, ini untuk menentukan prioritas dalam pemecahan masalah.



Sumber : http://www.bexcellence.org

Gambar 2.6 Contoh Diagram Pareto

Dr. Joseph Juran menemukan bahwa konsep ini secara universal bisa diterapkan di berbagai bidang. Dia mengkoinkan ucapan bagian vital dan memiliki banyak kegunaan.

Pareto diagram adalah grafik dengan mengklasifikasikan rangking data secara menurun dari kiri ke kanan. Pareto diagram digunakan untuk mengidentifikasikan banyak permasalahan yang penting.

Diagram Pareto ini dapat digunakan sebagai alat interpretasi untuk hal-hal berikut:

1. Menentukan frekuensi relative dan urutan pentingnya masalah-masalah atau penyebab-penyebab dari masalah yang ada.
2. Memfokuskan perhatian pada isu-isu yang kritis dan penting melalui pengurutan prioritas terhadap masalah-masalah atau penyebab-penyebab dari masalah yang ditemukan.

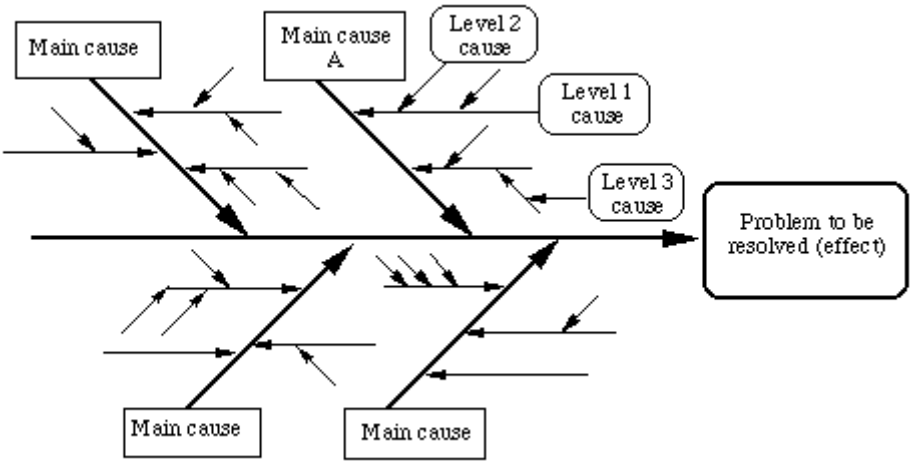
b. *Cause and Effect Diagram*

Dale H Besterfield (1994, p22) *Cause and effect* diagram adalah gambar yang terdiri dari garis dan simbol yang menggambarkan arti hubungan antara efek dan penyebab dari efek tersebut. CE ni ditumukan oleh Dr. Kaoru Ishikawa, diagram ini bisa disebut juga dengan Ishikawa diagram atau *fishbone*.

Bagaimana kegagalan kualitas terjadi? menurut Dr. Ishikawa kegagalan tersebut dapat terjadi karena metode kerja, material, pengukuran, manusia, dan lingkungan.

Dale H Besterfield (1994, PP24) Diagram sebab akibat berguna untuk:

1. Analisis keadaan yang terjadi meningkatkan kualitas produk atau pelayanan, dan meminimasi biaya.
2. Menghilangkan kondisi penyebab yang membuat produk tidak nyaman untuk konsumen sehingga konsumen mengkomplain.
3. Mengstandarisasikan keadaan sekarang dan yang akan datang.
4. Mengedukasikan dan melatih pekerja dalam mengambil keputusan dan mengkoreksi pergerakan dari aktivitas.



Sumber: ifm.eng.cam.ac.uk

Gambar 2.7 Contoh Cause and Effect Diagram

Pada dasarnya diagram sebab akibat dapat dipergunakan untuk kebutuhan-kebutuhan berikut :

i. Membantu mengidentifikasi akar penyebab dari suatu masalah.

j. Membantu membangkitkan ide-ide untuk solusi suatu masalah.

k. Membantu dalam penyelidikan ataupun pencarian fakta lebih lanjut.

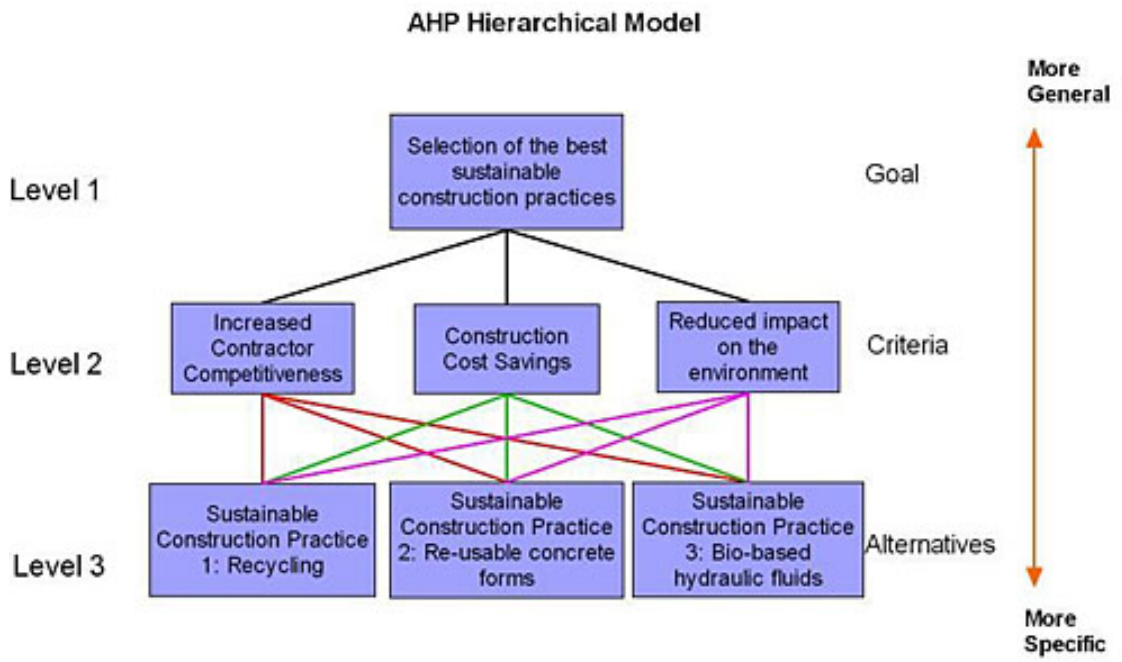
c. *Analytichal Hirearchy Process* (AHP)

AHP merupakan sebuah teknik terstruktur yang digunakan untuk menghadapi keputusan yang kompleks. AHP membantu pembuat keputusan untuk menemukan hasil yang paling sesuai dengan kebutuhan dan pengertian mereka terhadap suatu permasalahan dengan memecahnya secara hirarki ke dalam beberapa sub permasalahan dimana setiap sub

permasalahan akan dianalisis secara independen. Setiap elemen dari setiap hirearki dapat berkaitan dengan berbagai aspek dalam pengambilan keputusan.

Berikut merupakan langkah-langkah dalam membuat analisis AHP :

a. Memodelkan masalah ke dalam sebuah hirearki yang mengandung tujuan, alternatif-alternatif untuk mencapai tujuan,dan kriteria-kriteria untuk mengevaluasi alternatif tersebut.



Sumber: http://www. lifecyclebuilding.org

Gambar 2.8 Hirearki dalam AHP

b. Menetapkan prioritas diantara elemen-elemen dalam hirearki dengan membuat sekumpulan penilaian berdasarkan perbandingan berpasangan dari masing-masing elemen.

Tabel 2.2 Skala Penilaian Perbandingan Berpasangan pada AHP

The Fundamental Scale for Pairwise Comparisons		
Intensity of Importance	**Definition**	**Explanation**
1	Equal importance	Two elements contribute equally to the objective
3	Moderate importance	Experience and judgment slightly favor one element over another
5	Strong importance	Experience and judgment strongly favor one element over another
7	Very strong importance	One element is favored very strongly over another; its dominance is demonstrated in practice
9	Extreme importance	The evidence favoring one element over another is of the highest possible order of affirmation
Intensities of 2, 4, 6, and 8 can be used to express intermediate values. Intensities 1.1, 1.2, 1.3, etc. can be used for elements that are very close in importance.		

Sumber: Gaspersz (2002, p 137)

c. Mengumpulkan penilaian-penilaian untuk menghasilkan kumpulan prioritas untuk hirearki.

Tabel 2.3 Matriks Penilaian Alternatif

Kriteria 1				Kriteria 2				Kriteria 3			
Faktor	A	B	C	Faktor	A	B	C	Faktor	A	B	C
A				A				A			
B				B				B			
C				C				C			

Sumber: Gaspersz (2002, p 138)

d. Menguji konsistensi dari penilaian

Penilaian dinyatakan konsisten jika nilai dari *Consistency Ratio* lebih kecil atau sama dengan 1. Berikut ini merupakan rumus-rumus yang dipakai untuk mendapatkan nilai *Consistency Ratio*:

Consistency Index (CI) = $= \frac{\lambda - n}{n-1}$

Dimana n adalah banyaknya alternatif, dan λ adalah rata-rata dari *Consistency Vector*.

Consistency Ratio (CR) = $= \frac{CI}{RI}$

Dimana RI merupakan *Random Index* yang didapatkan dari tabel RI.

Tabel 2.4 Random Index

Matrik Order	1	2	3	4	5	6	7	8	9	10	11
Ramdom Index	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51

Sumber: Gaspersz (2002, p 138)

e. Menghasilkan keputusan akhir berdasarkan hasil AHP

2.2.2.4. Fase *Improve*

Fase *improve* merupakan fase yang berguna untuk menghasilkan desain, ide, dan implementasi perbaikan serta validasi perbaikan. Hal terpenting dalam fase improve adalah proses *brainstorming*, menganalisis *Failure Mode and Effect Analysis*, analisis awal *cost/benefit*, dan rekomendasi perbaikan

informasi yang berhubungan serta siapa yang bertanggung jawab dalam memantau dan mengendalikan proses perbaikan kualitas ini.

berikut ini adalah tools yang dapat digunakan dalam mendukung fase *improve* yaitu:

a. *Failure Mode and Effect Analysis*

FMEA adalah suatu cara di mana suatu bagian atau suatu proses yang mungkin gagal memenuhi suatu spesifikasi, menciptakan cacat atau ketidak sesuaian dan dampaknya pada pelanggan bila mode kegagalan itu tidak dicegah atau dikoreksi (Brue, 2002, p130).

Metode FMEA mempunyai banyak aplikasi dalam lingkungan *Six sigma*, untuk mencari berbagai masalah bukan hanya dalam proses serta perbaikan kerja, tapi juga dalam aktivitas pengumpulan data prosedur serta pelaksanaan *Six sigma*. Prasyarat yang diperlukan adalah dengan memberikan penekanan khusus untuk menghentikan masalah. Konsep kunci penggunaan FMEA adalah :

1. Mendaftarkan masalah-maslah potensial yang dapat muncul.
2. Menilai masalah dengan menggunakan skala 1-10 untuk setiap kegagalan potensial untuk 3 kategori berikut :

   - *Occurance* (O), suatu perkiraan probabilitas atau peluang bagi penyebab akan terjadi dan menghasilkan modus kegagalan yang menyebabkan akibat tertentu.

Tabel 2.5 Skala *Occurrence*

Skala	Kriteria Verbal	Tingkat Kejadian
1	Tidak mungkin penyebab ini mengakibatkan kegagalan	1 dalam 1000000
2 3	Kegagalan akan jarang terjadi	1 dalam 20000 1 dalam 4000
4 5 6	Kegagalan agak mungkin terjadi	1 dalam 1000 1 dalam 400 1 dalam 80
7 8	Kegagalan adalah sangat mungkin terjadi	1 dalam 40 1 dalam 20
9 10	Hampir dapat dipastikan bahwa kegagalan akan terjadi	1 dalam 8 1 dalam 2

- *Severity* (S), suatu perkiraan subyektif bagaimana buruknya pengguna akhir akan merasakan akibat dari kegagalan tersebut

Tabel 2.6 Skala *Severity*

Skala	Kriteria Verbal
1	*Neglible Severity*, kita tidak perlu memikirkan akibat ini akan berdampak pada kinerja produk. Pengguna akhir tidak akan memperhatikan kecacatan ini.
2 3	*Mild Severity*, akibat yang ditimbulkan hanya bersifat ringan, pengguna akhir tidak merasakan perubahan kinerja.
4 5 6	*Moderate Severity*, pengguna akhir akan merasakan akibat penurunan kinerja atau penampilan namun masih berada dalam batas toleransi.
7 8	*High Severity*, pengguna akhir akan merasakan akibat buruk yang tidak dapat diterima, berada di luar batas toleransi.
9 10	*Potential Safety Problem*, akibat yang ditimbulkan adalah sangat berbahaya dan bertentangan dengan hukum.

- *Detectibility* (D), perkiraan subyektif bagaimana efektivitas dan metode pencegahan atau pendeteksian.

Tabel 2.7 Skala *Detectability*

Skala	Kriteria Verbal	Tingkat Kejadian
1	Metode pencegahan atau deteksi sangat efektif. Tidak ada kesempatan bahwa penyebab akan muncul lagi.	1 dalam 1000000
2 3	Kemungkinan bahwa penyebab itu terjadi adalah sangat rendah.	1 dalam 20000 1 dalam 4000
4 5 6	Kemungkinan penyebab bersifat moderat, Metode deteksi masih memungkinkan kadang kadang penyebab itu terjadi.	1 dalam 1000 1 dalam 400 1 dalam 80
7 8	Kemungkinan penyebab itu masih tinggi. Metode pencegahan kurang efektif, penyebab masih berulang lagi	1 dalam 40 1 dalam 20
9 10	Kemungkinan penyebab itu terjadi sangat tinggi. Metode deteksi tidak efektif. Penyebab akan selalu terjadi	1 dalam 8 1 dalam 2

3. *Risk Priority Number* (RPN) merupakan hasil perkalian antara skala *severity*, *detectibility* dan skala *occurance*, untuk memprioritaskan kegagalan potensial.

$$RPN = O \times S \times D$$

Melakukan tindakan-tindakan untuk mengurangi resiko kegagalan, dengan memfokuskan pada kegagalan potensial yang memiliki nilai RPN (prioritas) tertinggi.

2.2.2.5. Fase *Control*

Fase *control* adalah fase terakhir dari metode *DMAIC*, dalam fase ini dilakukan pengaturan proses atau perbaikan produk serta pemantauan kinerja yang sedang berjalan. Selain itu, pada fase c*ontrol* juga memastkan bahwa perbaikan yang baru dapat dilakukan. Rencana pengendalian dapat berjalan baik ketika perusahaan mendokumentasikan semua.

2.3 Sistem Informasi

2.3.1 Pengertian Sistem

Menurut Mathiassen et al. (2000, p9), sistem adalah sekumpulan komponen yang mengimplementasikan persyaratan *model*, *function* dan *interface.*

O'Brien (2003, p8), mengatakan bahwa) sistem yaitu sebuah kelompok yang terintegrasi dan bekerja sama untuk mencapai tujuan yang sama dengan menerima masukan *(input)* dan menghasilkan keluaran *(output)* dalam sebuah proses transformasi yang terorganisir dengan baik.

Sedangkan menurut R. McLeod (2001, p11) Sistem adalah sekelompok elemen-elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan.

Berdasarkan bentuk sumber daya yang membentuk sistem, sistem dapat dibagi menjadi dua jenis, diantaranya adalah sebagai berikut :

a. Sistem fisik (*conceptual system*), yaitu sistem yang terbentuk dari sumber daya fisik. Perusahaan adalah salah satu contoh sistem fisik.

b. Sistem konsep (*conceptual system*), yaitu sistem yang menggunakan sumber daya konsep untuk menggambarkan sistem fisik. Sumber daya konsep terdiri dari informasi dan data.

2.3.2 Pengertian Informasi

Menurut O'Brien (2003, p13), informasi adalah data yang telah diolah menjadi bentuk yang bermakna dan berguna bagi pengguna akhir. Perbedaan dari data dengan informasi adalah, data merupakan fakta-fakta yang belum mengalami pengolahan, sedangkan informasi merupakan data yang telah diolah dalam bentuk yang teratur, dan terorganisasi dengan baik, sehingga dapat berguna oleh penerima informasi. Informasi dibutuhkan karena informasi merupakan suatu hal yang dapat dijadikan pertimbangan dalam pengambilan keputusan.

Menurut McLeod, 2001, p15 Informasi adalah data yang telah diproses, atau data yang memiliki arti dan siap dipakai. Informasi juga bisa diartikan sebagai data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya.

Informasi diperlukan karena informasi merupakan suatu dasar dalam mengambil keputusan. Kualitas dari informasi dapat ditentukan oleh empat dimensi McLeod (2001, p145), yaitu :

- Relevansi: informasi yang didapat oleh pembuat keputusan harus mempunyai relevansi terhadap tanggung jawab dan tugas mereka.
- Akurasi: informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan, informasi harus jelas mencerminkan maksud dari sumber ke penerimanya sehingga

pembuat keputusan akan semakin terbantu dan yakin akan informasi yang diterimanya ketika harus membuat keputusan.

- Ketepatan waktu: informasi yang disediakan oleh sistem informasi dapat dipergunakan oleh orang yang tepat pada waktu yang tepat untuk mengambil keputusan, kebijakan, atau tindakan yang tepat.
- Kelengkapan: informasi yang diperoleh oleh pembuat keputusan harus sesuai dengan kebutuhan. Jika terlalu sedikit akan menyulitkan dalam membuat keputusan yang akurat dan tepat waktu. Jika terlalu banyak atau melebihi dari yang dibutuhkan atau dapat dipergunakan, pembuat keputusan seringkali mengabaikan informasi dari masalah yang serius.

2.3.3 Pengertian Sistem Informasi

Menurut O'Brien (2003, p7), sistem informasi adalah kombinasi terorganisir dari manusia, perangkat keras, perangkat lunak, jaringan telekomunikasi dan sumber daya data, yang mengumpulkan, mengubah dan mendistribusikan informasi di dalam organisasi.

Sistem informasi adalah suatu kombinasi yang terorganisasi dari manusia, perangkat lunak, perangkat keras, jaringan komunikasi dan sumber daya data yang mengumpulkan, mentransformasikan, serta menyebarkan informasi di dalam sebuah organisasi (Mcleod 2001, p4).

Sistem informasi merupakan suatu alat bantu yang dirancang untuk membantu tingkat manajemen organisasi dengan menyediakan informasi yang berguna di dalam pengambilan keputusan organisasi baik pada tingkat perencanaan strategis, perencanaan manajemen maupun perencanaan operasi untuk mencapai tujuan organisasi. Komponen-komponen dari sistem

informasi adalah metode kerja (*work practices*), informasi (*information*), manusia (*people*), dan teknologi informasi (*information technologies*).

2.3.4 Pengertian Analisis dan Perancangan Sistem

Menurut McLeod, 2001, p190 Analisis sistem adalah penelitian atas sistem yang telah ada dengan tujuan untuk merancang sistem yang baru atau diperbaiki. Dapat disimpulkan bahwa analisis sistem adalah penelitian sistem yang ada dengan tujuan penyempurnaan sistem yang dapat dimanfaat oleh pengguna.

Menurut McLeod (2001, p192) , perancangan sistem adalah penentuan proses dan data yang diperlukan oleh sistem baru. Perancangan sistem merupakan tindak lanjut dari analisis sistem, jika analisis sistem dilakukan dengan baik maka pelaksanaan perancangan sistem yang diusulkan akan menghasilkan sistem yang baik dan mampu mengatasi masalah-masalah yang dihadapi sistem lama tanpa menimbulkan suatu masalah baru.

Perancangan sistem adalah proses penerjemahan kebutuhan pengguna ke dalam alternatif rancangan sistem informasi yang diajukan kepada pengguna informasi untuk dipertimbangkan. Atau dengan kata lain, perancangan sistem merupakan suatu proses penyiapan spesifikasi dalam menerjemahkan kebutuhan pengguna dalam pengembangan sistem baru.

Perancangan sistem bertujuan untuk memenuhi kebutuhan pemakai sistem serta untuk memberikan gambaran yang jelas dan rancang bangun yang lengkap.

Langkah-langkah yang dilakukan dalam tahap perancangan sistem yaitu :

a. Menyiapkan rancangan sistem yang terinci

b. Mengidentifikasi berbagai alternatif konfigurasi sistem

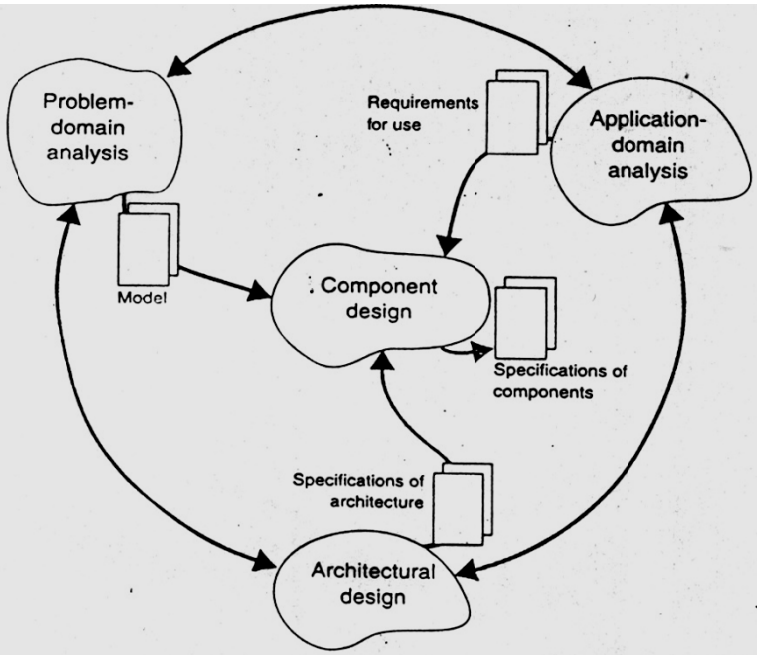
c. Mengevaluasi berbagai alternatif konfigurasi sistem

d. Memilih konfigurasi terbaik

e. Menyiapkan usulan penerapan

f. Menyetujui atau menolak penerapan sistem

2.3.5 *Object Oriented Analysis and Design*



Gambar 2.9 Aktivitas dan Hasil Utama *OOAD*

(Sumber : Mathiassen et al., 2000, p15)

Analisis dan perancangan berorientasi objek mempunyai empat aktivitas utama, yaitu *problem domain analysis*, *application domain analysis*, *architectural design*, dan *component design*.

2.3.5.1 *Object* dan *Class*

Menurut Mathiassen (2000, p4) Objek adalah sebuah entitas (*entity*) dengan identitas (*identity*), keadaan (*state*), dan tingkah laku (*behaviour*). Di dalam analisis, sebuah objek adalah

abstraksi dari fenomena di dalam konteks sistem. Objek mengekspresikan pandangan pengguna akan realitas. Di dalam desain atau perancangan, sebuah objek adalah bagian dari sistem.

Class adalah deskripsi dari sekumpulan objek yang berbagi struktur, *behavioural pattern*, dan atribut.

Analisis dan perancangan berorientasi objek mendeskripsikan dua permasalahan yang berbeda, yakni di dalam dan di luar sistem. Analisis objek mendeskripsikan fenomena di luar sistem, seperti orang dan barang, yang dapat berdiri sendiri. Perancangan objek mendeskripsikan fenomena di dalam sistem yang dapat diawasi. *Behavior* objek dapat dideskripsikan sebagai operasi untuk komputer yang menyelesaikannya.

2.3.5.2 *Model Context*

Problem domain adalah bagian dari konteks yang diadministrasikan, dimonitor, atau dikontrol oleh sistem. *Application domain* adalah organisasi yang mengadministrasikan, memonitor, dan mengontrol *problem domain*.

Problem domain mendeskripsikan tujuan sistem dan *application domain* merupakan bagian dari organisasi pengguna. Kesuksesan atau kegagalan sistem bergantung pada bagaimana menghubungkan *problem domain* dan *application domain* menjadi satu kesatuan secara fungsional.

Tugas utama dalam analisis dan perancangan adalah memodelkan sistem apa yang akan diadministrasikan, dimonitor, atau dikontrol. Selanjutnya adalah memodelkan bagaimana sistem akan berinteraksi dengan pengguna dalam *application domain*. Pengembangan sistem memerlukan pemahaman umum diantara pengembang dan pengguna. Analisis berorientasi objek

berfokus pada hal ini. pendekatan berorientasi objek berguna dalam semua tipe pengembangan sistem.

2.3.5.3 *System Definition*, Kriteria *FACTOR*, dan *Rich Picture*

Menurut Mathiassen (2000, p24) *System definition* merupakan deskripsi yang mengartikan banyak hal (*concise description*) dari sebuah sistem terkomputerisasi yang diekspresikan dalam bahasa sehari-hari. Hal ini menjelaskan sistem didalam konteks, informasi apa yang harus dipunyai, fungsi apa yang harus disediakan, dimana digunakannya, dan bagaimana kondisi pengembangannya.

Tujuan dari pendefinisian ini alah untuk menjelaskan berbagai interpretasi dan kemungkinan. *System definition* membantu memelihara pandangan dari opsi berbeda dan digunakan untuk membandingkan berbagai alternatif yang ditemui. *System definition* harus jelas dan tepat, dan mengandung banyak keputusan dasar mengenai sistem.

Salah satu cara menjelaskan *system definition* adalah menggunakan kriteria *FACTOR*, yaitu *Functionality, Application domain, Conditions, Technology, Objects,* dan *Responsibility.*

- ***Functionality***

  Merupakan fungsi sistem yang mendukung tugas dari *application domain*.

- ***Application domain***

  Merupakan bagian dari organisasi yang mengadministrasi, memonitor, atau mengontrol *problem domain.*

- ***Conditions***

  Merupakan kondisi dimana sistem akan dikembangkan dan digunakan.

- ***Technology***

Merupakan teknologi yang digunakan dalam mengembangkan sistem dan teknologi yang dibutuhkan untuk menjalankan sistem.

- ***Objects***

  Merupakan objek utama di dalam *problem domain*.

- **Responsibility**

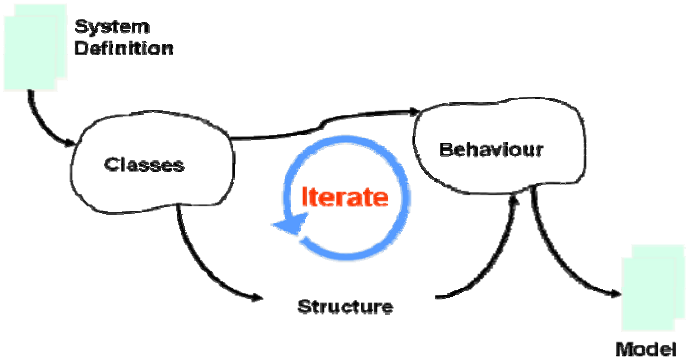
  Merupakan tanggung jawab sistem secara keseluruhan didalam konteks sistem.

Rich picture merupakan suatu penggambaran dari sistem yang membantu orang awam untuk mengerti keadaan dari sistem yang sedang berjalan maupun sistem yang akan diusulkan. Sebuah *rich picture* berfokus pada aspek-aspek penting dari keadaan yang berjalan, yang ditentukan sendiri oleh seorang *illustrator*. *Rich picture* harus mampu memberikan gambaran yang luas dari kondisi yang ada sehingga memungkinkan adanya beberapa penafsiran.

2.3.6 *Problem Domain Analysis*

(Mathiassen et al., 2000, p4) *Problem domain analysis* berfokus untuk menjawab pertanyaan “Dengan informasi apa sistem harus berhubungan?”. Selama aktivitas analisis, model *problem domain* menyediakan bahasa untuk mengekspresikan kebutuhan kepada sistem. Selama perancangan, model ditransformasikan menjadi sebuah komponen yang merepresentasikan kondisi *problem domain*. Model adalah deskripsi *class*, objek, struktur, dan *behaviour* di dalam *problem domain*.

Aktivitas dalam memodelkan *problem domain* terdiri dari tiga bagian, yaitu : pembuatan *class* dari *system definition*. Dari *class* akan dibuat *structure* dan *behaviour*, dimana *behaviour* juga didapatkan dari *structure*. Dari behavoiur maka akan terbentuk suatu model.



Gambar 2.10 Aktivitas utama *Problem Domain Analysis*

(Sumber : Mathiassen et al., 2000, p46)

2.3.6.1 *Class*

Menurut mathiassen (2000, p49) *Object* adalah suatu entitas dengan *identity* (identitas), *state* (pernyataan) dan *behavior* (perilaku). *Class* adalah suatu deskripsi dari sekumpulan objek yang mempunyai *structure, behavioral pattern* dan *attributes. Event* adalah kejadian terus-menerus yang melibatkan satu atau dua objek.

Abstraksi, klasifikasi, dan seleksi merupakan tugas utama dalam aktivitas *class*. Fenomena *problem domain* diabstraksikan dengan melihatnya sebagai *object* dan *event*. *Object* dan *event* ini kemudian diklasifikasikan dan dipilih manakah *class* dan *event* yang akan dipelihara informasinya oleh sistem. *Class* adalah yang pertama kali didefinisikan dan dibatasi dalam *problem domain*. *Class* akan dideskripsikan karakteristiknya dengan sekumpulan *event* yang spesifik.

Aktivitas *class* ini akan menghasilkan *event table*, dimana dimensi horizontal

menunjukkan *class* yang dipilih, dan dimensi vertikal menunjukkan *event* yang dipilih. Tanda '+' mengindikasikan bahwa objek dari *class* terlibat pada *event* tertentu sebanyak nol atau satu kali. Sedangkan tanda '*' mengindikasikan bahwa objek dari *class* terlibat pada *event* tertentu sebanyak nol sampai banyak kali.

Kandidat bagi *class* adalah berupa kata benda (*noun*) dan merupakan tipe general. Penamaan *class* haruslah sederhana dan mudah dibaca, berasal dari *problem domain*, mengindikasikan sebuah *instance*. Sementara kandidat sebuah *event* adalah berupa kata kerja dan merupakan tipe general. Penamaan *event* pun haruslah sederhana dan mudah dibaca, berasal dari *problem domain*, mengindikasikan sebuah *event*.

(Mathiassen et al., 2000, p60) Cara mengevaluasi kriteria *class* dan *event* adalah dengan menemukan apakah *class* dan *event* tersebut berada di dalam *system definition* dan relevan bagi model *problem domain*. *Class* haruslah dapat diidentifikasikan objek-objeknya, mengandung informasi yang unik, menunjukkan banyak objek, dan berhubungan dengan sejumlah *event*. Sedangkan evaluasi bagi kandidat *event* adalah bila *event instantaneous*, *atomic*, dan dapat diidentifikasikan kapan terjadinya.

2.3.6.2 *Structure*

(Mathiassen et al., 2000, p69) *Structure* bertujuan untuk mendeskripsikan hubungan struktural diantara *class* dan objek di dalam *problem domain*. Konsep *structure* terbagi menjadi dua, yaitu *class structure* dan *object structure*. Hasil dari *structure* adalah sebuah *class* diagram dengan *class* dan struktur.

Class structure menggambarkan hubungan konseptual yang statis antar *class*, terdiri dari g*eneralization* dan *cluster*. *Generalization* merupakan suatu hubungan antara satu atau lebih sub-

class dengan satu atau lebih *superclass.* Dan c*luster* merupakan kumpulan dari *class* yang saling berhubungan.

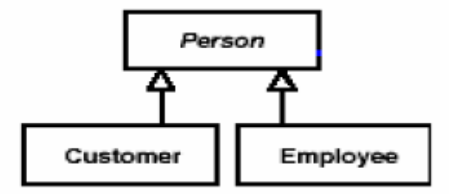
Object structure menggambarkan hubungan yang dinamis antara objek yang ada dalam *problem domain*, terdiri dari *agregation* dan *association*. *Aggregation* mendefinisikan hubungan antara dua buah objek atau lebih, dimana *superior* objek (*the whole*) terdiri dari sejumlah objek (*the part*). *Association* merupakan hubungan yang bermakna diantara sejumlah objek.

Cara menemukan kandidat bagi *structure* adalah dengan mempelajari *abstract*, hubungan statis diantara *class* dan mempelajari *concrete*, hubungan dinamis diantara objek.

Aktifitas *structure* difokuskan pada bagaimana suatu *class* dan *object* dihubungkan. Aktivitas ini kemudian akan menghasilkan sebuah *class diagram*. Mathiassen (2000, p72) mengklasifikasikan hubungan struktural menjadi 2 yaitu struktur antar kelas dan struktur antar objek .

Adapun struktur antar kelas meliputi :

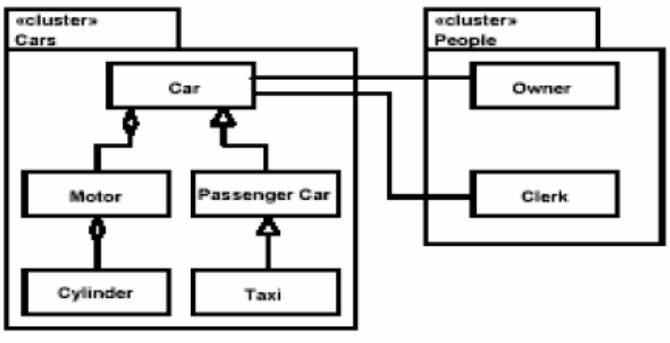
a. **Generalisasi**: merupakan hubungan struktural antara dua atau lebih kelas yang khusus (*subclass*) dengan sebuah *class* yang umum (*super class*) dimana semua properti yang terdapat pada *superclass* juga terdapat pada *subclass*. Struktur generalisasi seringkali didefinisikan sebagai hubungan "*is a*".



sumber : Mathiassen, et *al.* (2000, p73)

Gambar 2.11 Contoh Struktur Generalisasi

b. ***Cluster***: merupakan sebuah kumpulan dari *classes* yang berhubungan. Kelas didalam *Cluster* biasanya berhubungan secara struktur generalisasi atau struktur agregasi.

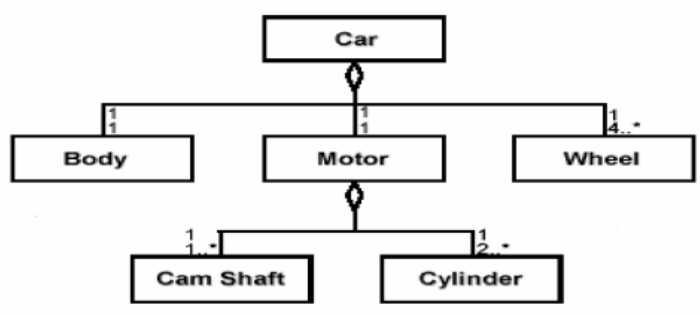


sumber : Mathiassen, et *al.* (2000, p75)

Gambar 2.12 Contoh Struktur *Cluster*

Dan struktur antar objek meliputi :

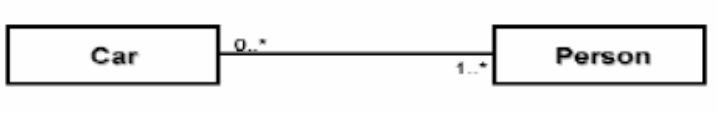
a. **Agregasi:** merupakan hubungan struktural diantara dua atau lebih objek yang merupakan bagian dari sebuah objek lainnya. Struktur agregasi seringkali didefinisikan sebagai hubungan "*has a*".



sumber : Mathiassen, et *al.* (2000, p76)

Gambar 2.13 Contoh Struktur Agregasi

b. **Asosiasi**: Struktur asosiasi adalah sebuah hubungan antara dua atau lebih objek dimana kedua objek tersebut sejajar dan tidak mendefinisikan objek lainnya.



sumber : Mathiassen, et *al.* (2000, p77)

Gambar 2.14 Contoh Struktur Asosiasi

(Mathiassen et al., 2000, p78) Terdapat tiga tipe aplikasi dari struktur *aggregation* :

1. *Whole part*, dimana keseluruhan adalah merupakan penjumlahan dari bagian-bagiannya; jika ditambahkan atau dihapus satu bagian maka akan mengubah keseluruhan secara mendasar.

2. *Container content*, dimana keseluruhan adalah *container* dari bagian-bagiannya; jika ditambahkan atau dihapus satu *content* maka properti dari keseluruhan tidak akan berubah secara mendasar.

3. *Union member*, dimana keseluruhan adalah merupakan *union* anggota yang diorganisasikan. Dengan menambah atau menghapus sebagian kecil anggota, *union* tidak akan berubah secara mendasar. Bagaimanapun, terdapat batas terbawah pada sejumlah anggota, yang merupakan pendukung bagi model sebuah *union* tanpa anggota.

(Mathiassen et al., 2000, p80) Pola merupakan deskripsi umum dari masalah dan solusi yang berhubungan. Pola dapat diterapkan untuk memecahkan masalah khusus selama *problem-domain-modelling*. Terdapat empat pola yaitu, *role pattern*, *relation pattern*, *hierarchy pattern*, dan *item-decriptor pattern*.

Role pattern digunakan untuk memodelkan situasi dimana *single person* dapat memiliki beberapa peran dalam *problem domain*. Tujuan dari *relation pattern* adalah untuk menghubungkan dua bagian agar saling terkait. *Hierarchy pattern* adalah dimana objek pada satu

level dapat memiliki beberapa objek pada level dibawahnya. *Item-decriptor pattern* berguna dalam sistem untuk mengadministrasikan tipe deskripsi yang berbeda, seperti kontak, kebijakan asuransi, dan spesifikasi produk.

Struktur dievaluasi dengan kriteria bahwa struktur harus benar, terkonseptual, dan sederhana.

2.3.6.3 *Behaviour*

(Mathiassen et al., 2000, p89) Tujuan *behavior* adalah untuk memodelkan *problem domain* yang dinamis. Hasil dari *behavior* adalah sebuah *behavior pattern* dengan atribut pada setiap *class* dalam *class* diagram. Tiga konsep yang terkandung dalam *behavior* adalah :

- *Event trace*, merupakan urutan dari *events* yang melibatkan objek secara spesifik.
- *Behavioral pattern*, merupakan suatu deskripsi dari kemungkinan *event trace* untuk semua objek dalam *class.*
- *Attribute*, merupakan suatu deskripsi dari *class* atau *event.*

(Mathiassen et al., 2000, p90) *Behaviour*al *pattern* adalah deksripsi kemungkinan *event trace* bagi seluruh objek didalam sebuah *class*. *Behaviour*al *pattern* digambarkan dalam notasi *statechart* diagram.

Behaviour dalam setiap *class* yang ditunjukkan dalam *event table* harus konsisten dengan *statechart* diagram (Mathiassen et al., 2000, p100).

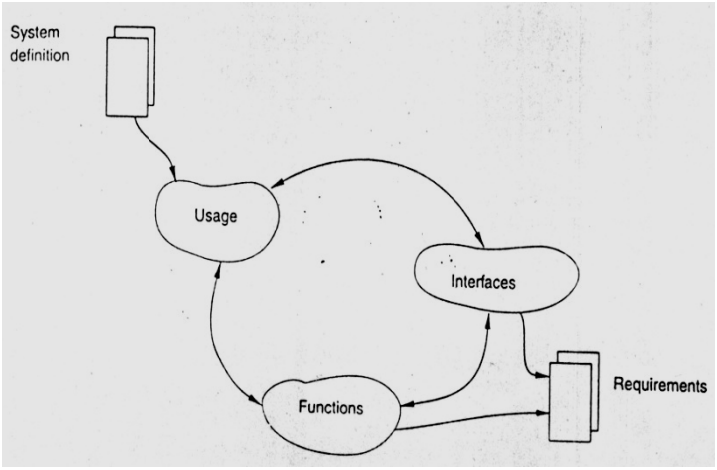
(Mathiassen et al., 2000, p94) *Statechart* diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari rangsangan yang diterima. Pada umumnya *statechart* diagram menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart* diagram). *Statechart* diagram menunjukkan urutan-

urutan *state* dari sebuah objek selama masa hidupnya dan *event-event* yang menyebabkan perubahan *state* tersebut.

(Mathiassen et al., 2000, p341) *State* digambarkan berbentuk segi empat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dan dapat dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

2.3.7 *Application Domain Analysis*

Menurut Mathiassen (2000, p115) *Application domain analysis* yaitu organisasi yang mengadministrasi, memonitor atau mengontrol sebuah *problem domain*. Tujuannya adalah untuk mendapatkan sebuah daftar lengkap kebutuhan *usage* dari sistem. Aktivitas *application domain analysis* adalah u*sage, function,* dan *interface.*



Gambar 2.15 Aktivitas dalam Analisis *Application Domain*

(Sumber : Mathiassen et al., 2000, p117)

2.3.7.1 *Usage*

(Mathiassen et al., 2000, p119) *Usage* bertujuan untuk menentukan bagaimana aktor berinteraksi dengan sistem. Dua konsep *usage* adalah *actor* dan *use case*. Prinsip dari *usage* adalah menentukan *application domain* dengan *use case*, mengevaluasi *use case* dalam kolaborasinya dengan pengguna, dan menilai perubahan sosial dalam *application domain*.

Actor adalah sebuah abstraksi dari pengguna atau sistem lain yang berinteraksi dengan *target system. Use case* adalah urutan kejadian-kejadian antara sistem dan aktor dalam *application domain. Actor* dan *use case* dideskripsikan dalam *actor specification* dan *use case specification.*

2.3.7.2 *Function*

(Mathiassen et al., 2000, p137) *Function* merupakan fasilitas untuk membuat sebuah model berguna bagi aktor. Tujuan *function* adalah untuk menentukan kemampuan pemrosesan informasi dari sistem. Prinsip dari *function* adalah mengidentifikasikan semua *function*, menspesifikasikan *function* yang kompleks, mengecek konsistensi antara *use case* dengan model. Hasil dari *function* adalah sebuah daftar lengkap dari *function* (*function list*) dengan spesfikasi *function* yang kompleks.

Function list adalah daftar fungsi-fungsi yang ada dalam sistem yang dideskripsikan tingkat kompleksitas dan tipe fungsinya. Empat tipe *function*, diantaranya adalah:

- ***Update function***

  Function yang diaktifkan dengan *event* pada *problem domain* dan menghasilkan perubahan pada *state* model.

- ***Signal function***

Function yang diaktifkan dengan mengubah *state* model dan menghasilkan reaksi di *context*. Reaksi ini mungkin menampilkan aktor pada *application domain* atau berintervensi langsung di *problem domain.*

- ***Read function***

  Function yang diaktifkan oleh kebutuhan akan informasi di lembar kerja aktor dan hasilnya tampilan sistem yang relevan dari model.

- ***Compute function***

  Function yang diaktifkan oleh kebutuhan akan informasi di lembar kerja aktor melibatkan informasi yang disediakan aktor atau model. Hasilnya adalah tampilan dari kegiatan *compute* tersebut.

2.3.7.3 *Interface*

(Mathiassen et al., 2000, p151) *Interface* adalah fasilitas yang membuat model sistem dan *function* dapat digunakan oleh aktor. Tujuan *interface* adalah untuk menetapkan *interface* sistem. Hasil dari *interface* adalah *user* dan *system interface*.

User interface merupakan tipe dialog dan *form* presentasi, daftar lengkap dari elemen *user interface*, *window* diagram dan *navigation* diagram. Sedangkan *system interface* merupakan *class* diagram untuk peralatan luar dan protokol-protokol untuk berinteraksi dengan sistem lain.

Berikut ini adalah penjabaran dari *Interfaces* yang terdiri dari *user interface* dan *system interface*:

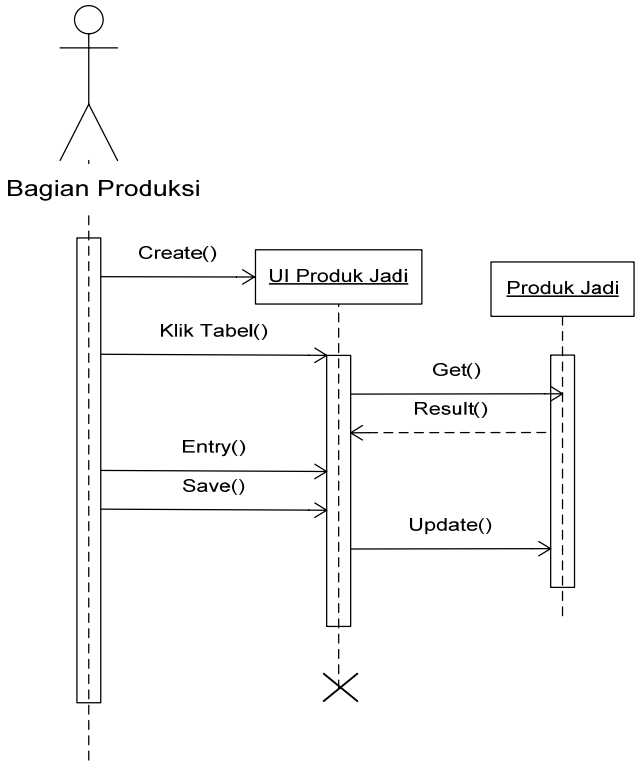
a) ***User interface***

   Merupakan *style dialog* dan bentuk-bentuk presentasi, daftar elemen dari *user interface* yang lengkap, *windows diagram* yang dipilih dan *navigation diagram.*

b) ***System interface***

merupakan *class diagram* untuk eksternal *device* dan protokol-protokol untuk interaksi dengan sistem lain. *Navigation diagram* merupakan semua window dari *user interface* dan hubungan dinamiknya.

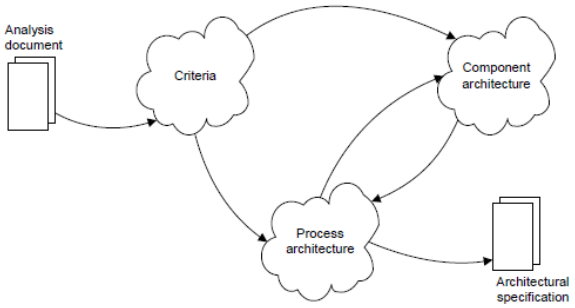
Hasil dari aktivitas *interface* pada *application domain* meliputi *user interface* dari sistem dan *sequence diagram* untuk menggambarkan interaksi-interaksi yang terjadi di dalam suatu *user interface*. Selain itu, pada aktivitas ini juga dihasilkan *Navigation Diagram* yang menggambarkan *window-window* yang terdapat dalam sistem dan transisi yang terjadi diantara *window-window* tersebut (Mathiassen, 2000, p344)



Gambar 2.16 Contoh *Sequence* Diagram

2.3.8 *Architectural Design*

(Mathiassen et al., 2000, p173) *Architectural design* bertujuan untuk menstrukturkan sistem yang terkomputerisasi. Prinsip *architectural design* adalah mendefinisikan dan memprioritaskan *criteria*, menjembatani *criteria* dengan *technical platform*, dan mengevaluasi desain lebih awal. Hasil dari *architectural design* adalah struktur dari komponen dan proses sistem. Tiga aktivitas dari *architectural design* adalah *criteria*, *component architecture*, dan *process architecture*.



Gambar 2.17 Aktivitas dalam *Architectural Design*

(Sumber : Mathiassen et al., 2000, p176)

2.3.8.1 *Criteria*

(Mathiassen et al., 2000, p177) *Criteria* bertujuan untuk mengatur prioritas perancangan, dengan hasil sekumpulan prioritas kriteria. Prinsip *criteria* adalah perancangan baik (*good design*) yang tidak memiliki kelemahan, seimbang diantara beberapa *criteria*, *usable*, *flexible*, dan *comprehensible*. Konsep *criteria* adalah *criterion*, yang merupakan properti dari *architecture*, dan *condition*, yaitu kesempatan dan batas *technical*, *organizational* dan *human* yang telibat dalam suatu tugas.

(Mathiassen et al., 2000, p178) Dua belas kriteria klasik dari kualitas *software*, yaitu :

- *Usable*

Kemudahan di dalam implementasi sistem yang dikembangkan.

- *Secure*

  Keamanan akan otoritas pemakaian data dalam sistem.

- *Efficient*

  Sistem yang dirancang memiliki sifat efisiensi dari eksploitasi ekonomis.

- *Correct*

  Terpenuhinya kebutuhan/persyaratan sistem.

- *Reliable*

  Sistem yang dikembangkan dapat memenuhi hasil dari eksekusi fungsi-fungsi yang ada.

- *Maintainable*

  Sistem yang dirancang harus dapat dilakukan *maintenance*

- *Testable*

  Sistem yang dibuat dapat diuji coba.

- *Flexible*

  Sistem bersifat fleksibel dalam pemanfaatan sistem yang dirancang.

- *Comprehensible*

  Usaha yang dibutuhkan untuk mendapatkan pemahaman yang jelas dari sistem.

- *Reusable*

  Potensial penggunaan dari bagian sistem dengan sistem lain yang berhubungan.

- *Portable*

  Sistem dapat digunakan dari satu sistem ke sistem *platform* yang lainnya.

- *Interoperable*

  Sistem dapat dioperasikan dengan sistem lainnya.

Menurut Mathiassen (2000, p185) Prioritas kriteria desain mendeskripsikan prioritas dari setiap kriteria, yaitu *very important, important, less important, irrelevant,* atau *easily fulfilled.*

2.3.8.2 *Component Architecture*

Menurut Mathiassen (2000, p189) *Component* bertujuan untuk menciptakan sistem yang *comprehensible* dan *flexible.* Prinsip *component* adalah mengurangi kompleksitas, merefleksikan struktur konteks yang stabil, dan menggunakan kembali komponen yang telah ada. Hasil dari *component* adalah sebuah *class* diagram dengan spesifikasi dari komponen kompleks.

Component architecture adalah sebuah struktur sistem dari *component* yang saling berhubungan. *Component* adalah kumpulan bagian program yang menggambarkan keseluruhan dan didefinisikan dengan baik *responsibility*-nya.

Pola *architectural* adalah *layered architectural pattern, generic architectural pattern,* dan *client-server architectural pattern*. *Component* terdiri dari tiga bagian, yaitu model (M), *function* (F), dan *interface* (UI).

Komponen dari suatu sistem terdiri dari :

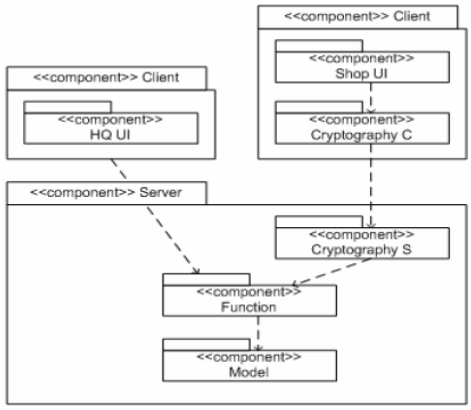
c) Model – bertanggung-jawab menampung objek-objek yang menggambarkan *problem domain.*
d) Function – bertanggung-jawab dalam menyediakan fungsi-fungsi dari sistem
e) User Interface – bertanggungjawab untuk menangani interaksi antara pengguna dengan sistem

Dalam komponen diagram dapat menggambarkan distribusi dalam *client server architecture*:

Tabel 2.8 Client Server Architecture

Client	*Server*	*Architecture*
U	U + F + M	*Distributed presentation*
U	F + M	*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)



sumber : Mathiassen, et *al.* (2000, p201)

Gambar 2.18 Contoh *Component Diagram*