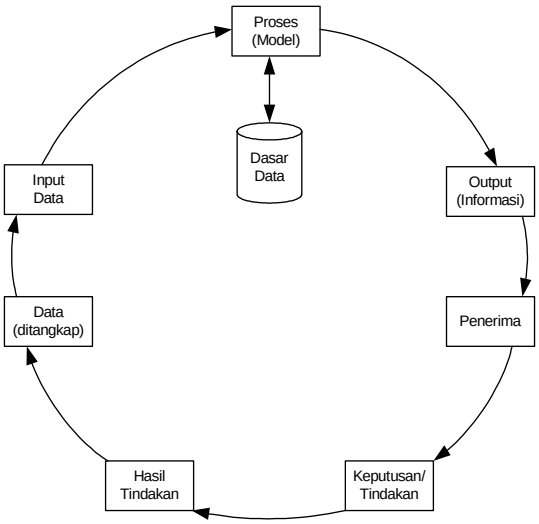
Seterusnya membentuk suatu siklus. Siklus ini disebut siklus informasi (*information cycle*) dan disebut juga dengan siklus pengolahan data (*data processing cycle*).




**Gambar 2.1.** Siklus Informasi

[ Sumber : Analisis dan Desain Sistem Informasi, Jogiyanto Hartono MBA., Ph.D., Penerbit Andi Yogyakarta ( 2001) ]

### 2.1.2 Nilai Informasi

Nilai dari informasi (*value of information*) ditentukan oleh dua hal, yaitu manfaat dan biaya mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibanding dengan biaya mendapatkannya. Akan tetapi perlu diperhatikan bahwa informasi yang digunakan di dalam suatu sistem informasi umumnya digunakan untuk beberapa kegunaan, sehingga sulit untuk menghubungkan suatu bagian informasi pada suatu masalah tertentu dengan perbandingan biaya untuk memperolehnya, karena sebagian besar informasi digunakan tidak hanya oleh satu pihak di dalam perusahaan.

Informasi tidak dapat percis ditaksir dengan satuan uang, tetapi dapat ditaksir dengan efektivitasnya. Pengukuran nilai informasi biasanya dihubungkan dengan analisis *cost effectiveness* atau *cost benefit*.

## 2.2 Pengertian Sistem

Didalam suatu sistem informasi terdapat dua kelompok pendekatan yang biasa digunakan dalam mendefinisikan sistem, yaitu pendekatan yang menekankan pada prosedur dan yang menekankan pada komponen atau elemennya.

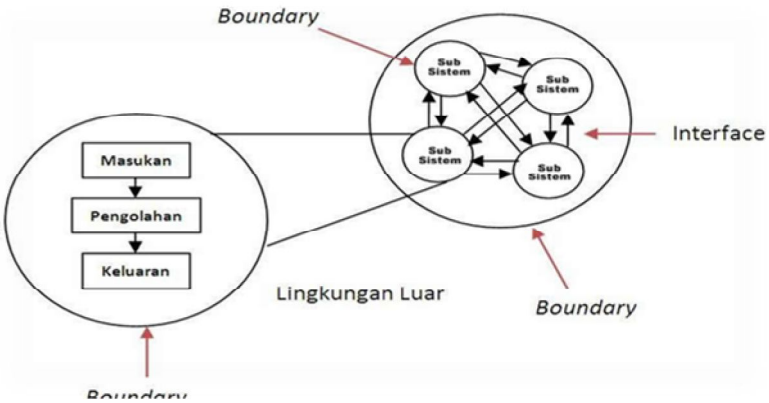
Pendekatan sistem yang lebih menekankan pada prosedur, mendefinisikan sistem sebagai berikut [HM, Jogiyanto 3]:

*"Suatu sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berkaitan, berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu".*

Sedangkan pendekatan sistem yang lebih menekankan pada komponen atau elemen, mendefinisikan sistem yaitu [HM, Jogiyanto 3] :

*"Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu".*

Sistem itu sendiri memiliki karakteristik atau beberapa sifat tertentu, yaitu mempunyai komponen (*components*), batas sistem (*boundary*), lingkungan luar sistem (*environments*), penghubung (*interface*), masukan (*input*), keluaran (*output*), pengolah (*process*), dan sasaran (*objectives*) atau suatu tujuan (*goal*).




**Gambar 2.2**. Karaktersitik Sistem

[ Sumber : Jogiyanto, H. M., MBA, Ph.D., 2001, Analisis & Desain Sistem, Andi Yogyakarta, Yogyakarta ]

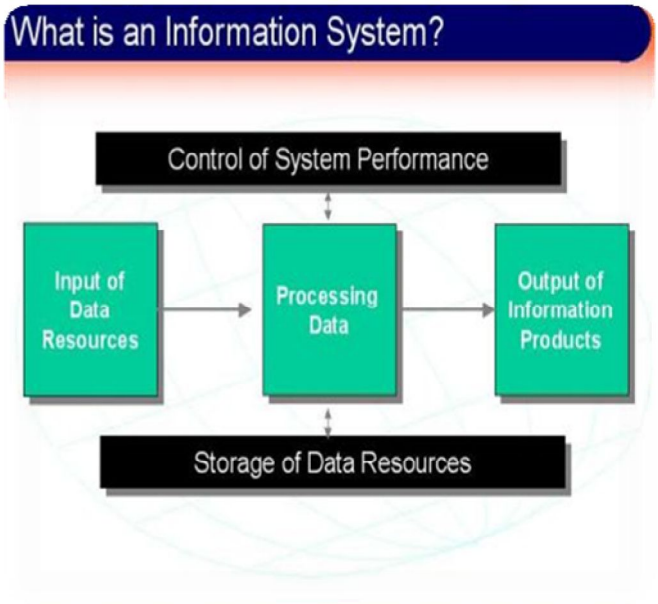
## 2.3 Pengertian Sistem Informasi

Informasi merupakan hal yang sangat penting di dalam mendukung pengambilan keputusan. Informasi dapat diperoleh dari sistem informasi (*Information* sistem) atau disebut juga dengan *processing* sistem atau *information processing* sistem atau *information generating* sistem. Sistem informasi didefinisikan oleh Robert A. Leitch dan K. Roscoe Davis di daam buku Jogiyanto (2001:11) sebagai berikut:

*"sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan yang diperlukan".*

Sistem informasi sendiri memiliki sejumlah komponen tertentu. Seperti yang dikemukakan oleh Robert dan Donald Symanzky dalam buku Jogiyanto (2001:12), bahwa sistem informasi terdiri dari beberapa komponen yang berbeda yaitu, manusia, data, *hardware*, dan *software*. Sebagai suatu sistem, setiap komponen tersebut

berinteraksi satu dengan lainnya membentuk satu kesatuan untuk mencapai sasarannya.




**Gambar 2.3**. Bagan Sistem Informasi

[ Sumber http://blog.its.ac.id/dyah03tc/*file*s/2007/10/sistem-informasi.jpg

Time Download : 16/10/09 at 1:17 PM ]

## 2.4 Pengembangan dan Pendekatan Perancangan Sistem Informasi

.Pengembangan sistem (sistem *development*) dapat berarti menyusun sistem yang baru untuk menggantikan sistem yang lama secara keseluruhan atau memperbaiki sistem yang telah ada. Sistem yang lama perlu diganti atau diperbaiki disebabkan karena beberapa hal, yaitu :

1. Adanya permasalahan-permasalahan (*Problems)* yang timbul di dalam sistem lama. Permasalahan-permasalahan yang timbul dapat berupa :

   a. Ketidakberesan

Ketidakberesan dalam sistem yang lama menyebabkan sistem yang lama tidak dapat beroperasi sesuai dengan yang diharapkan. Ketidakberesan ini dapat berupa :

1. Kecurangan-kecurangan yang disengaja yang menyebabkan tidak amannya harta kekayaan perusahaan dan kebenaran dari data menjadi tidak terjamin.
2. Kesalahan-kesalahan yang tidak disengaja yang juga dapat menyebabkan kebenaran dari data tidak terjamin.
3. Tidak efisiennya operasi.
4. Tidak ditaatinya kebijaksanaan manajemen yang telah ditetapkan.

b. Pertumbuhan Organisasi

Pertumbuhan organisasi menyebabkan harus disusunnya sistem yang baru. Pertumbuhan organisasi diantaranya adalah kebutuhan informasi yang semakin luas, *volume* pengolahan data semakin meningkat, perubahan prinsip akuntansi yang baru. Karena adanya perubahan ini menyebabkan sistem yang lama tidak efektif lagi, sehingga sistem yang lama sudah tidak dapat memenuhi lagi kebutuhan informasi yang dibutuhkan manajemen.

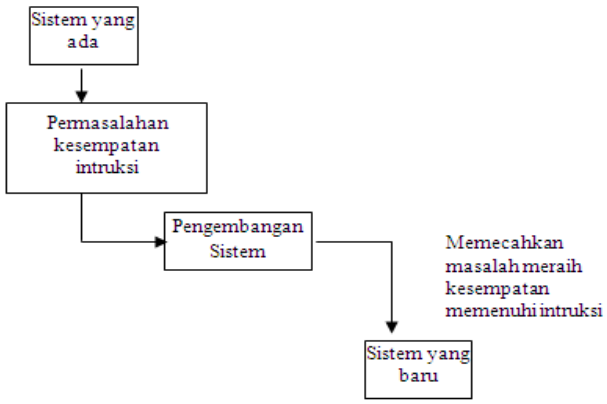
2. Untuk meraih kesempatan-kesempatan (*Opportunities*)

Teknologi informasi telah berkembang dengan cepatnya. Organisasi mulai merasakan bahwa teknologi informasi ini perlu digunakan untuk meningkatkan

penyediaan informasi sehingga dapat mendukung dalam pengambilan keputusan. Kecepatan informasi atau efisiensi waktu sangat menentukan berhasil atau tidaknya strategi dan rencana yang telah disusun untuk meraih kesempatan-kesempatan yang ada.

3. Adanya intruksi-intruksi (*Directives*)

Pengembangan sistem yang baru dapat juga terjadi karena adanya intruksi-intruksi dari atas pimpinan ataupun dari luar organisasi, seperti misalnya peraturan pemerintah.




**Gambar 2.4.** Pengembangan Sistem Informasi

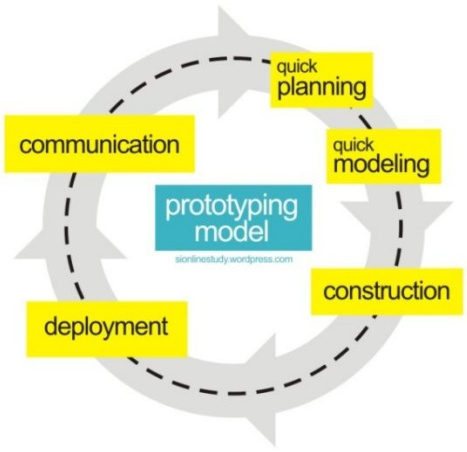
[ Sumber : Analisis dan Desain Sistem Informasi, Jogiyanto Hartono MBA., Ph.D., Penerbit Andi Yogyakarta (2001) ]

Terdapat beberapa metode pendekatan dan perancangan sistem informasi yang digunakan oleh berbagai pengembang sesuai dengan tujuan dari pengembagan sistem informasi yang dihadapi, dibawah ini merupkan pendekatan dan perancangan sistem informasi yang populer digunakan oleh banyak *developer.*

### 2.4.1 Pengembangan Sistem Informasi Metode *Prototype*

Metode pengembangan sistem yang digunakan adalah menggunakan model *prototype,* karena model ini dianggap cocok dengan pendekatan beorientasi objek. Metode ini memungkinkan pemakai ikut serta dalam menentukan kebutuhan dan menentukan sistem apa yang akan di kerjakan untuk memenuhi kebutuhan tersebut.

*Prototype* di gunakan untuk mengembangkan kebutuhan pemakaian yang sulit di definisikan untuk melancarkan proses SDLC, *Prototype* paling baik di gunakan untuk mengembangkan sistem yang kurang di definikan kurang baik dan cocok untuk menerapkan sistem kecil dan unik.




**Gambar 2.5 Metode Prototype**

[ Sumber : Pressman, S. Roger, 2001 ]

### 2.4.2 Pendekatan Sistem Berorientasi Objek

Metodologi (analisis dan perancangan perangkat lunak) berorientasi objek diusulkan oleh penulis-penulis buku/artikel kemudian praktek-praktek penerapannya oleh para penulis tersebut terlembagakan, contoh-contoh metode /metodologi :

- Shlaer/Mellor
- Coad/Yourdon
- Object Modelling Techniques (Rumbaugh, et. al.)
- Responsibility Driven Design (Wirfs-Brock, et. al.)
- Objectory
- OOAD Booch
- Fusion (Coleman, et. al.)

Tiap metode/metodologi berorientasi objek memiliki teknik dan notasi sendiri yang satu sama lain berbeda Kesamaan umum dari semua metode/metodologi berorientasi objek terletak pada cara pandang, yaitu **:**

a. **Static,** menggambarkan keterkaitan logik antar entitas pada sistem

b. **Dynamic**, menggambarkan hubungan yang terbentuk akibat dinamika sistem.

### 2.4.2.1 Konsep *Objek Oriented Analisis And Desain*(*OOAD*)

Dalam dunia pemodelan, metodologi implementasi obyek walaupun terikat kaidah-kaidah standar, namun teknik pemilihan obyek tidak terlepas pada subyektifitas *software analist* & *designer*. Beberapa obyek akan diabaikan dan beberapa obyek menjadi perhatian untuk diimplementasikan di dalam sistem. Hal ini sah-sah saja karena kenyataan bahwa suatu permasalahan sudah tentu memiliki lebih dari satu solusi.

Object Oriented Analysis Design (OOAD) adalah rekayasa perangkat lunak yang model pendekatan sistem sebagai kelompok interaksi objek. Beberapa entitas mewakili setiap obyek, yang menarik dalam sistem menjadi model dan kelas, elemen data, dan perilaku. Analisis berorientasi objek (OOA) berlaku objek modeling teknik untuk menganalisis fungsional persyaratan untuk sistem.

Secara singkat OOAD adalah metode analisis yang memerikasa requirements dari sudut pandang kelas-kelas dan objek yang ditemui dalam ruang lingkup permasalahan yang mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem. Sekarang ini terdapat berbagai metodologi yang digunakan dalam object oriented analisys and design (OOAD).

Beberapa konsep dasar dalam OOAD adalah :

**a. Objek (Object)**

Objek adalah benda yang secara fisik dan konseptual yang ada disekitar kita. Beberapa contoh objek, seperti hardware, software, dokumen, manusia, konsep, dan lainnya. State adalah objek yang mempunyai keadaan sesaat yang menggambarkan keadaan dari objek tersebut. State dinyatakan dengan atribut, behaviour, interface, service, dan method.

**b. Kelas (Class)**

adalah definisi umum dari himpunan objek yang sejenis. Kelas menetapkan spesifikasi perilaku (behaviour) dan atribut-atribut dari objek tersebut. Class adalah abstraksi dari entitas dunia nyata.

**c. Kotak hitam (black Box)**

adalah dasar dari implementasi objek. Dalam operasi OO, hanya para developer yang dapat memahami detail dari proses-proses yang ada dalam kotak hitam, sedangkan para user tidak mengetahui. Menggunakan proses enkapsulasi dan message.

**d. Asosiasi dan Agregasi**

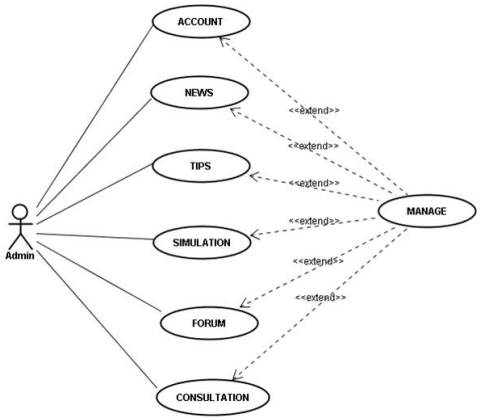
Asosiasi adalah hubungan yang mempunyai makna antara sejumlah objek. Asosiasi digambarkan dengan sebuah garis penghubung di antara objeknya. Agregasi adalah bentuk khusus sebuah asosiasi yang menggambarkan seluruh bagian pada suatu objek merupakan bagian dari objek yang lain.

### 2.4.2.2 Pengenalan UML

*Unified Modelling Language* (*UML*) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. UML mendefinisikan diagram-diagram berikut ini :

***1. Use case diagram***

*Use case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem.




**Gambar 2.6** Contoh *Use Case*

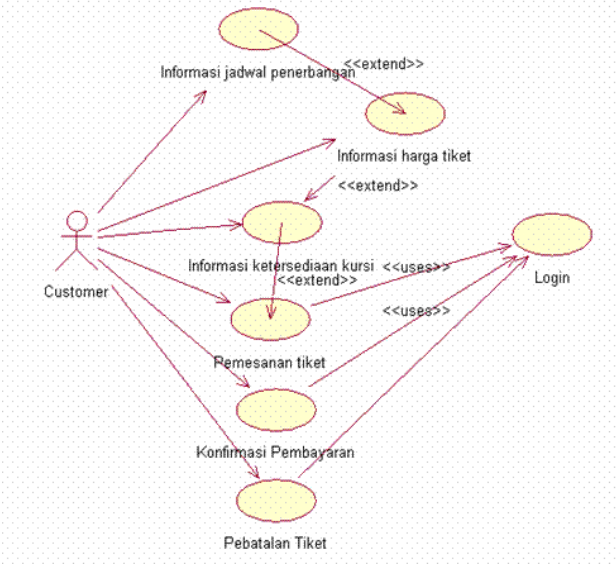
http://blog.its.ac.id/*dielife/files/2009/06/use_case2.JPG* 6 April 2010

### *2. Class diagram*

*Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda




**Gambar 2.7** Contoh *Class Diagram*

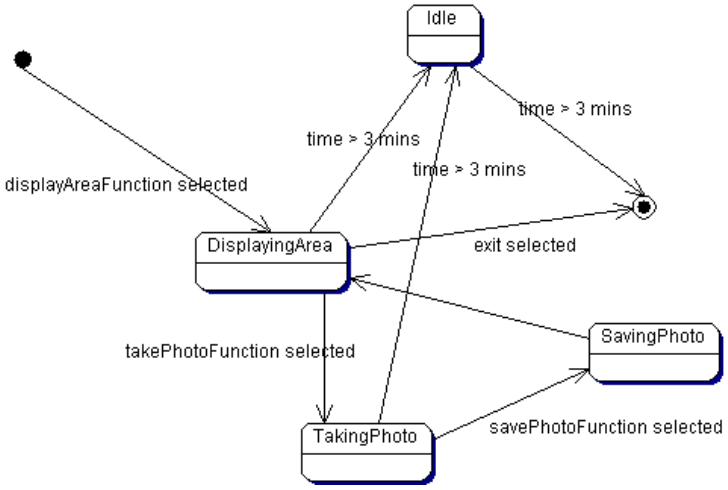
http://i761.photobucket.com/*albums/xx256/ditakumaira/use-case-diagram.gif* 6 April 2010

### 3. Behaviour Diagram

Terbagi kedalam 2 diagram yaitu :

#### *1. Statechart diagram*

Menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya *statechart* diagram menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).



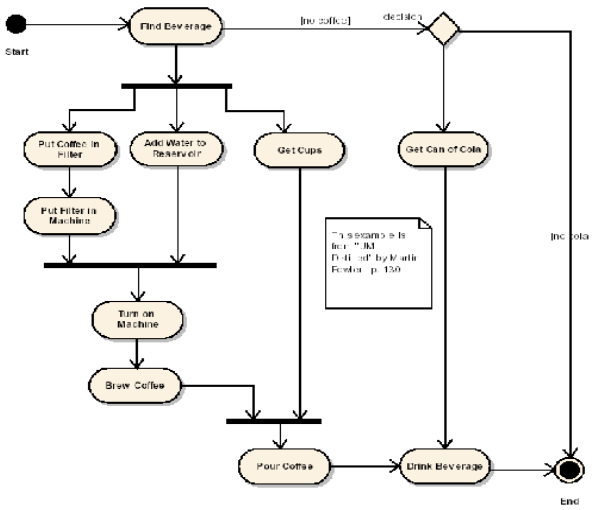

**Gambar 2.8** Contoh *Statechart Diagram*

http://vega.soi.city.ac.uk/~*aj406/XTraQue/PM1_ST_TakingAPhoto.gif*

6 April 2010

### *2. Activity Diagram*

*Activity diagram* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.



**Gambar 2.9** Contoh *Activity Diagram*

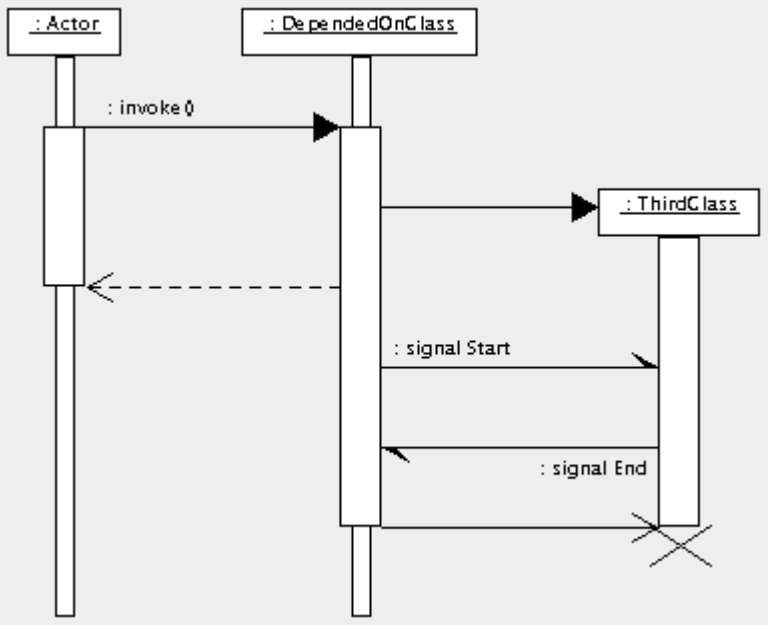
http://www.icn.ucl.ac.uk/courses/MATLABTutorials/Sessions2007_08/Christian_Kaul_3/UI%20Activity%20Diagram.jpg 6 April 2010

***4. Interaction diagram***

Terbagi menjadi 2 diagram, yaitu :

***1. Sequence Diagram***

*Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

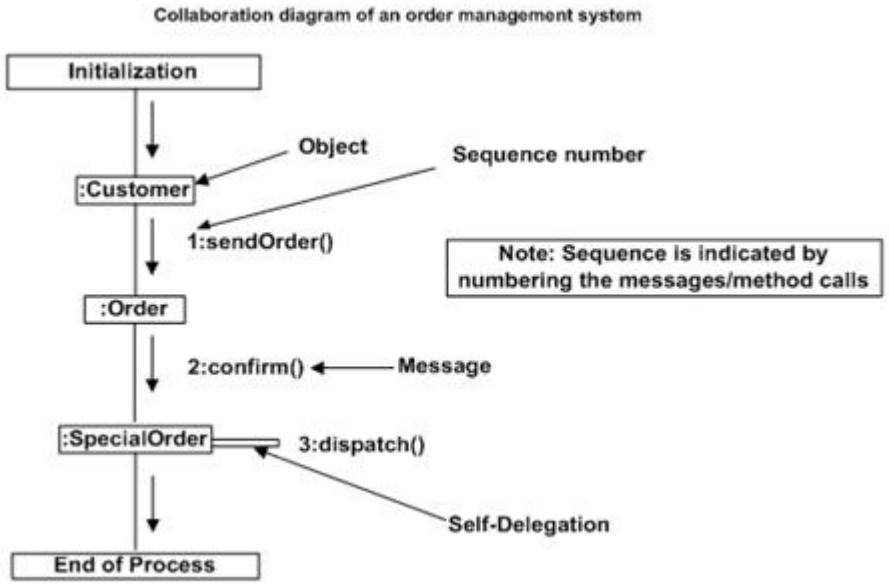



**Gambar 2.10** Contoh *Sequence Diagram*

http://argoumlstats.tigris.org/*documentation/printablehtml/manual/images/reference/sequence_diagram.gif* 6 April 2010

### *2. Collaboration Diagram*

*Collaboration diagram* juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message.*




**Gambar 2.11** Contoh *Collaboration Diagram*

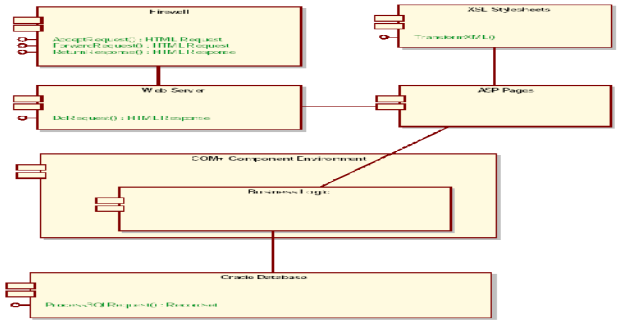
http://www.tutorialspoint.com/*images/uml_collaboration_diagram.jpg* 6 April 2010

### *5. Component diagram*

*Component diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya.

Komponen piranti lunak adalah modul berisi code, baik berisi source code maupun binary code, baik library maupun executable, baik yang muncul pada compile time, link time, maupun run time. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil.

Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

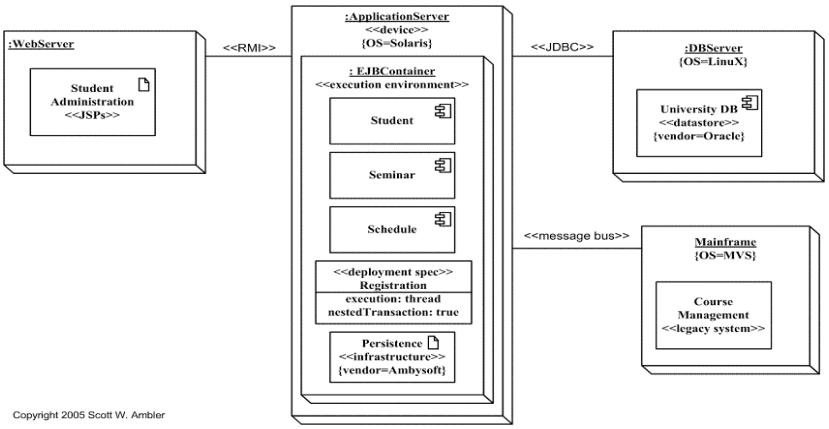



**Gambar 2.12** Contoh *Component diagram*

http://www.iist.unu.edu/*newrh/III/3/1/docs/rsltc/user_guide/html/UML2RSL_f17.png* 6 April 2010

### *6. Deployment diagram*

*Deployment/physical* diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.



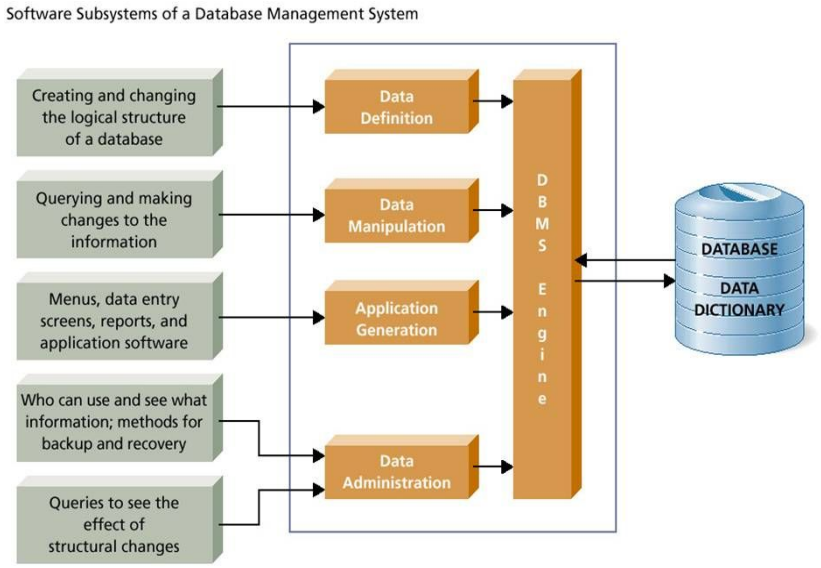

**Gambar 2.13** Contoh *Deployment diagram*

Scott W. Ambler (2005) from http://www.agilemodeling.com/images/style/deploymentDiagramProjectLevel.gif 6 April 2010

### 2.4.2.3 Konsep Dasar Basis Data

Secara sederhana *database* dapat diungkapkan sebagai suatu pengorganisasian data dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan cepat. Dalam hal ini, pengertian akses dapat mencakup pemerolehan data maupun pemanipulasian data seperti menambah serta menghapus data.

Sebagai mana diketahui, manajemen modern mengikutsertkana informasi sebagai sumber daya penting yang setara dengan sumber daya manusia, uang, mesin dan material.




**Gambar 2.14**. *RDBMS*

**Sumber : www.embeddedtechjournal.com/articles_2006/200... Time Download : 3/10/09 at 23:57 PM**

Dengan adanya komputer, data dapat disimpan dalam media pengingat yang disebu *hardisk.* Dengan menggunakan media ini kehadirian kerta yang digunakan dapat untuk menyimpan data dapat dikurangi. Selain itu, data menjadi lebih cepat untuk diakses, terutama dalam bentuk *database*

DBMS singkatan dari *Database Management* Sistem. DBMS merupakan perangkat lunak atau program komputer yang dirancang khusus untuk memudahkan pengkodean database. Salah satu macam DBMS yang popoluer dewasa ini berupa RDBMS dan ORDBMS , yang menggunakan model basis data atau dalam bentuk tabel-tabel yang saling berhubungan.

## 2.5 Jaringan Komputer

Jaringan komputer adalah sekumpulan komputer yang berhubungan antara yang satu dengan yang lainnya menggunakan *protocol* komunikasi sehingga dapat saling bertukar data dan informasi serta dapat menggunakan perangkat keras secara bersama. Selain itu jaringan komputer dapat diartikan sebagai sekumpulan terminal komunikasi yang berada di berbagai lokasi berbeda yang terdiri dari dua komputer atau lebih yang saling berhubungan (Budi Irawan (2005:6).

### 2.11.1 *Black-Box Testing*

Pada pengujian ini kita tidak perlu tahu apa yang sesungguhnya terjadi dalam sistem/perangkat lunak yang kita kembangkan. Kita menguji masukan dan keluarannya, artinya dengan berbagai masukan yang kita berikan, sistem/perangkat lunak memberikan keluaran yang kita harapkan.

Berdasarkan faktor uji :

1. *Reliability*

a) Menentukan toleransi.

b) Desain kontrol dan integritas data

c) Implementasi kontrol dan integritas data

d) Pengujian regresi, pengujian manual dan pengujian fungsional

e) Verifikasi dan ketepatan dan kelengkapan instalasi

f) *Update* ketepatan kebutuhan

2. *Authorization*

a) Identifikasi aturan otorisasi

b) Desain aturan otorisasi

c) Implementasi aturan otorisasi

d) Pengujian kesesuaian

e) Mencegah perrubahan data selam instalasi

f) Menjaga aturan otorisasi

3. *File Integrity*

a) Identifikasi kebutuhan integritas *file*

b) Desain kontrol dan integritas *file*

c) Implementasi kontrol dan integritas *file*

d) Pengujian fungsional

e) Verifikasi integritas dari produksi *file*

f) Menjaga integritas *file*

4. *Audit Trail*

a) Identifikasi kebutuhan rekonstruksi

b) *Desain audit trail*

c) Implementasi audit trail

d) Pengujian fungsional

e) Menyimpan *audit trail* selama instralasi

f) *Update audit trail*

5. *Continuity of processing*

a) Identifikasi akibat dari kegagalan

b) Desain *contingency plan*

c) Menyusun *contingency plan* dan prosedurnya

d) Pengujian pemulihan

e) Memastikan integritas dari pengujian sebelumnya

f) *Update contingency plan*

6. *Service Levels*

a) Identifikasi tingkat layanan yang diinginkan

b) Desain metode untuk mencapai tingkat layanan

c) Desain sistem untuk mencapai tingkat layanan

d) Pengujian beban lebih

e) Implementasi rencana pencegahan kegagalan instalasi

f) Menjaga tingkat layanan

7. Access kontrol

a) Identifikasi hak akses

b) Desain Prosedur akses

c) Implementasi proseddur keamanan

d) Pegujian kesesuaian

e) Kontrol akses selama instalasi

f) Menjaga keamanan

8. Metodology

a) Penyesuaian kebutuhan dengan metodology

b) Penyesuaian desain dengan metodology

c) Penyesuaian program dengan metodology

d) Penyesuaian pengujian dengan metodology

e) Penyesuaian integrasi dengan metodology

f) Penyesuaian perawatan dengan metodology

9. *Correctness*

a) Identifikasi spesifikasi fungsional

b) Penyesuaian desain dengan requirement

c) Penyesuaian program dengan desain

d) Pengujian fungsional

e) Ketepatan penempatan program dan data pada produksi

f) *Update* kebutuhan

10. *Ease of use*

a) Identifikasi spesifikasi kegunaan

b) Desain penggunaan fasilitas

c) Penyesuaian program dengan desain

d) Pengujian dukungan panduan

e) Penyebaran kegunaan instruksi

f) Menjaga kemudahan penggunaan

11. *Maintainable*

a) Identifikasi spesifikasi kegunaan

b) Desain dapat dirawat

c) Program dapat dirawat

d) Inspeksi

e) Kelengkapan dokumentasi

f) Menjaga keremawatan

12. *Portable*

a) Identifikasi kebutuan protabilitas

b) Desain protabilitas

c) Penyesuaian program dengan desain

d) Disaster testing

e) Kelengkapan dokumentasi

f) Menjaga protabilitas

13. *Coupling*

a) Identifikasi antar muka sistem

b) Kelengkapan desain antarmuka

c) Penyesuaian program dengan desain

d) Pengujian fungsional dan regresi

e) Koordinasi antarmuka

f) Memastikan antarmuka yang benar

14. *Performance*

a) Identifikasi kriteria performa

b) Kriteria pencapaian desain

c) Kriteria pencapaian program

d) Pengujian kesesuaian

e) Mengawasi performa instalasi

f) Menjaga tingkat performa

15. *Ease of operations*

a) Identifikasi kebutuhan operasional

b) Mengkomunikasikan kebutuhan pada operasi

c) Mengembangkan prosedur operasi

d) Pengujian operasi

e) Implementasi prosedur operasi