MODUL PRAKTIKUM
SIMULASI DAN PEMODELAN

Oleh :
Dadang Priyanto
NIDN : 08251174

UNIVERSITAS BUMIGORA
Juni 2022

LEMBAR VERIFIKASI DIKTAT

Modul mata kuliah praktikum Simulasi dan Pemodelan ini dilakukan verifikasi agar dapat dipergunakan untuk kegiatan proses belajar mengajar (PBM) terutama pada matakuliah Simulasi dan Pemodelan Program Studi S1 Rekayasa Perangkat Lunak (RPL) pada UNIVERSITAS BUMIGORA. Dengan memperhatikan isi materi pada Diktat ini, kami pihak laboratorium menyatakan bahwa;

1. Isi materi pada modul ini telah sesuai dengan silabus matakuliah
2. Tidak terdapat tumpang tindih materi yang diberikan terhadap materi lain

Terhadap kedua hal tersebut dinyatakan bahwa modul praktikum ini telah dapat dipergunakan untuk kegiatan praktikum dilaboratorium.

Terverifikasi, Mataram, 12/06/2022
Kaprodi S1 Rekayasa Perangkat Lunak

Tomi Tri Sujaka, S.Kom., M.Kom
NIK. 21.6.386

Modul 1
Analisis Sistem

A. Tujuan

- Mahasiswa mampu menjelaskan pengertian analisis sistem.
- Mahasiswa mampu menjelaskan tahapan analisis sistem
- Mahasiswa mampu menjelaskan analisis kebutuhan sistem dalam rekayasa perangkat lunak

B. Dasar Teori

Analisis sistem adalah kegiatan untuk mendefinisikan kebutuhan terkait sistem yang akan dikembangkan. Orang yang bertugas menganalisis sistem disebut sebagai *system analist*.

a. Tahapan Analisis Sistem

Di dalam melakukan analisis sistem tahapan yang dilalui adalah :

1. *IDENTIFY* → mengidentifikasi masalah
2. *UNDERSTAND* → memahami kerja sistem yang ada
3. *ANALYZE* → menganalisis sistem
4. *REPORT* → membuat laporan hasil

b. Tujuan dari analisis sistem antara lain :

- Menjabarkan kebutuhan pemakai
- Meletakkan dasar-dasar untuk proses perancangan perangkat lunak
- Mendefinisikan semua kebutuhan pemakai sesuai dengan lingkup kedua belah pihak

1. Analisis Kebutuhan Fungsional (*Functional Requirement*)

Merupakan kegiatan dalam mendeskripsikan fungsionalitas atau layanan yang diharapkan akan diberikan oleh sistem. Persyaratan fungsional untuk sistem perangkat lunak bisa dinyatakan dalam sejumlah cara. Persyaratan tersebut bergantung pada jenis perangkat lunak, user yang menggunakan, dan jenis sistem yang akan dikembangkan.

Contoh :

Terdapat sejumlah persyaratan fungsional untuk sistem perpustakaan (Kotonya dan Sommerville, 1998) bagi mahasiswa dan dosen untuk memesan buku dan dokumen dari perpustakaan lain :

1. User dapat mencari semua atau satu set awal database atau memilih subset darinya.
2. Sistem akan menyediakan viewer yang sesusia bagi user untuk membaca dokumen pada penyimpanan (store) dokumen.

Pada prinsipnya spesifikasi persyaratan fungsional untuk sebuah sistem harus lengkap dan konsisten.

2. Analisis Kebutuhan Non Fungsional (*Non Functional Requirement*)

Merupakan persyaratan yang tidak langsung berhubungan dengan fungsi spesifikyang disediakan oleh sistem. Persyaratan ini mungkin berhubungan dengan properti sistem yang muncul belakangan seperti kehandalan, waktu tanggap, dan penempatan pada media penyimpanan. Alternatifnya, persyaratan ini dapat mendefinisikan batasan pada sistem seperti kemampuan pirnti I/O dan representasi data yang dipakai pada interface sistem.

Kebutuhan non fungsional ini meliputi kebutuhan :

a. ***Development Requirement***
   Tools yang digunakan (hardware dan software) untuk pengembangan sistem.
   Contoh : eclipse, netbeans, starUML dsb.
b. ***Deployment Requirement***
   Terkait dengan lingkungan dimana sistem akan digunakan.
   Contoh: sistem harus mampu berjalan dengan spesifikasi RAM 4GB, OS Ubuntu, dsb.
c. ***Performance Requirement***
   Terkait dengan ukuran kualitas maupun kuantitas khususnya terkait dengan kecepatan, skalabilitas, dan kapasitas.
   Contoh: sistem harus mampu diakses oleh 100org dalam waktu bersamaan.
d. ***Dokumentation Requirement***
   Terkait dengan dokumen apa saja yang akan disertakan pada produk akhir.
   Contoh : dokumen teknis (dokumen perencanaan proyek, analisis , desain, pengujian), user manual,dan dokumen pelatihan.
e. ***Support Requirement***
   Kebutuhan yang terkait dengan dukungan yang diberikan setelah sistem informasi digunakan.
   Contoh: perlu adanya pelatihan bagi calon pengguna.

C. Praktikum

SKENARIO:

Minimarket milik pak joko menjual mulai peralatan rumah tangga, alat tulis, dan barang kelontong untuk kebutuhan sehari-hari. Pak joko ingin membuat sistem yang bisa digunakan untuk transaksi jual beli. Sistem tersebut akan digunakan oleh Pak Joko sebagai direktur dan karyawannya sebagai kasir. Jika anda diminta membangun sistem tersebut, analisislah kebutuhan sistemnya terlebih dahulu.

Step 1 : Identifikasi objek (entitas) lengkap dengan atribut yang terlibat di dalam sistem tersebut
Objek = direktur, barang, kasir

Step 2 : Tuliskan jawaban anda kedalam model kamus data
Kamus data minimarket :
Direktur = {nama, alamat, no telp}
Barang = {kode_barang, nama_barang, jenis, stok}
Kasir = {kode_kasir, nama, alamat, no_telp}

Step 3 : Tuliskan kebutuhan fungsional sistem
1. Kasir dapat menggunakan sistem untuk transaksi penjualan
2. Sistem dapat digunakan untuk menyimpan data barang baru
3. Direktur dapat melihat laporan data barang dan data penjualan

Step 4 : Tuliskan kebutuhan non fungsional sistem
1. Sistem ini berbasis desktop
2. Sistem dibangun dengan menggunakan bahasa pemrograman J2SE
3. Sistem dapat berjalan pada komputer dengan spesifikasi minimal RAM 1GB

D. Tugas Praktikum

1. Tentukan objek data dan atribut yang bisa ditemukan dalam apotek. Tuliskan hasil analisis kebutuhan fungsional dan nonfungsional untuk sistem informasi apotek.
2. Analisislah kembali dengan mengembangkan ide-ide baru pada sistem infomasi akademik (SIA) di kampus anda. Buatlah kebutuhan fungsional dan non fungsionalnya. Tuliskan teknologi baru yang anda gunakan, jika ada.

Modul 2
Data Flow Diagram (DFD)

A. Tujuan

- Mengenalkan DAD (Diagram Alir Data) sebagai tool perancangan system.
- Mahasiswa dapat menggunakan Microsoft Visio untuk menggambar DFD.

B. Dasar Teori

2.1 Microsoft Visio

Microsoft Visio (atau sering disebut Visio) adalah sebuah program aplikasi komputer yang digunakan untuk merancang suatu model perencanaan, model ini dimanfaatkan untuk kebutuhan developer maupun engineering yang didesain untuk berbagai macam kebutuhan. Microsoft Visio sering digunakan untuk membuat diagram, diagram alir (*flowchart*), brainstorm, dan skema jaringan yang dirilis oleh Microsoft Corporation. Aplikasi ini menggunakan grafik vektor untuk membuat diagram-diagramnya.

Gambar 2.1 Lambang Microsoft Visio

Visio, merupakan software yang mendukung untuk mendeskripsikan data dan fungsi/proses yang terlibat dalam proses rekayasa suatu perangkat lunak. Adapun data dan fungsi yang dimodelkan dalam proses dalam rekayasa perangkat lunak adalah

1. Pemodelan Data
   a. ERD (Diagram Keterhubungan antar Objek Data)
      Entity Relationship Diagram merupakan diagram yang dapat digunakan untuk melakukan aktifitas pemodelan data dan menggambarkan hubungan antara objek data.
      Dalam ERD terdapat tiga komponen yang digunakan yaitu entitas, atribut dan relasi.
      Contoh :
      - Entitas (Dosen)
      - Atribut : NIP, Nama Dosen, Alamat, No_Telpon, Jabatan

   b. Kamus Data
      Merupakan catatan yang digunakan untuk menyimpan deskripsi atau atribut dari semua objek data yang didefinisikan.
      Contoh : Dosen : { NIP, Nama Dosen, Alamat, No_Telpon, Jabatan }
2. DFD
   Mendeskripsikan seluruh fungsi yang terlibat dalam Perangkat Lunak. DFD atau Diagram Aliran Data, merupakan gambaran bagaimana data ditransformasikan pada sebuah sistem.

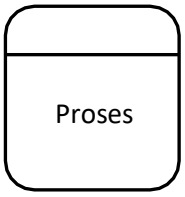
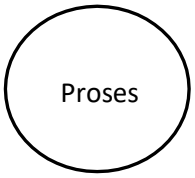
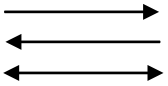
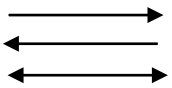
2.2 DFD

DAD (Diagram Aliran Data) atau yang juga dikenal dengan sebutan DFD (*Data Flow Diagram*) merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yg mudah dikomunikasikan oleh profesional sistem kepada pemakai

maupun pembuat program. Pada umumnya, DFD digunakan untuk merancang sistem yang menggunakan *data store* dalam mengelola informasi dalam sistem.

Komponen DFD, menurut Yourdan/DeMarco dan Gane/Sarson adalah sebagai berikut :

Tabel 2.2.2.1 Komponen DFD

Gane/Sarson	Yourdan/De Marco	Keterangan
		Entitas Eksternal, dapat berupa orang/unit terkait yang berinteraksi dengan sistem tetapi diluarsistem.
		Orang, unit yang mempergunakan atau melakukan transformasi data. Komponen fisik tidak diindetifikasikan.
		Aliran data dengan arah khusus dari sumber ke tujuan.
		Penyimpanan data atau tempat data dituju oleh proses

Keterangan :

a. Entitas Luar : kesatuan diluar sistem yang akan memberikan input atau menerima output dari sistem, dapat berupa orang atau,organisasi,sumber informasi lain atau penerima akhir dari suatu laporan.

b. Proses adalah transformasi input menjadi output, merupakan kegiatan atau pekerjaan yang dilakukan oleh orang atau mesin komputer, dimana aliran data masuk ditransformasikan ke aliran data keluar. Penamaannya sesuai dengan proses yang sedang dilakukan.

Ada beberapa hal yang perlu diperhatikan tentang proses :

1. Proses harus memiliki input dan output.
2. Proses dapat dihubungkan dengan komponen entitas luar, data store atau proses melalui alur data.
3. Sistem/bagian/divisi/departemen yang sedang dianalisis oleh profesional sistem digambarkan dengan komponen proses.
4. Penomoran proses dapat dilihat pada tabel berikut.

Tabel 2.2.2.2 Penomoran Proses

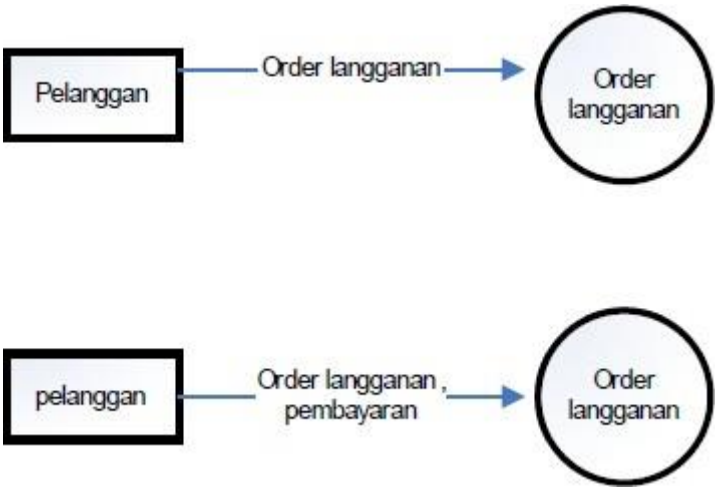
Nama Level	Nama Diagram	Nomor Proses
0	Konteks	0
1	Diagram level 1	1.0, 2.0, 3.0
2 2 2	Diagram rinci 1.0 Diagram rinci 2.0 Diagram rinci 3.0	1.1, 1.2, 1.3 2.1, 2.2, 2.3 3.1, 3.2, 3.3
3 3 3	Diagram rinci 1.1 Diagram rinci 1.2 Diagram rinci 1.3	1.1.1, 1.1.2, 1.1.3 1.2.1, 1.2.2, 1.2.3 1.3.1, 1.3.2, 1.3.2
dst		

c. **Aliran data/Arus data** digunakan untuk menjelaskan perpindahan data atau paket data dari satu bagian ke bagian lain.
Aliran data dapat berbentuk sebagai berikut:
- Formulir atau dokumen yang digunakan perusahaan
- Laporan tercetak yang dihasilkan sistem
- Output dilayar komputer
- Masukan untuk komputer
- Komunikasi ucapan
- Surat atau memo
- Data yang dibaca atau direkam di file
- Suatu isian yang dicatat pada buku agenda
- Transmisi data dari suatu komputer ke komputer lain

Catatan : aliran data tidak dalam bentuk kalimat.

Konsep Arus Data

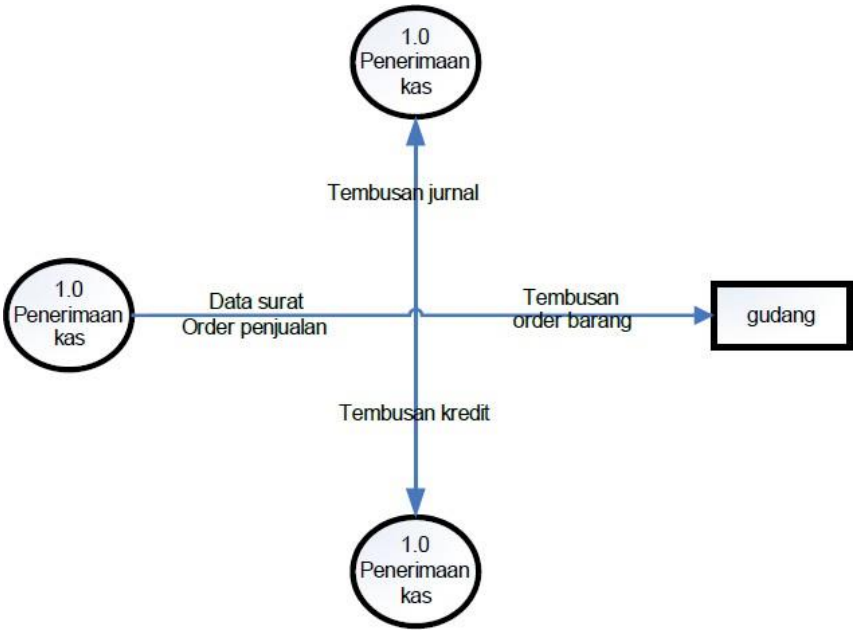
1. **Packet of Data (Paket Data)**
Bila dua data mengalir dari suatu sumber yang sama menuju ke tujuan yang sama, maka harus dianggap sebagai suatu arus data yang tunggal.



Gambar 2.2 Paket Data

2. Diverging Data Flow (Arus Data Menyebar)

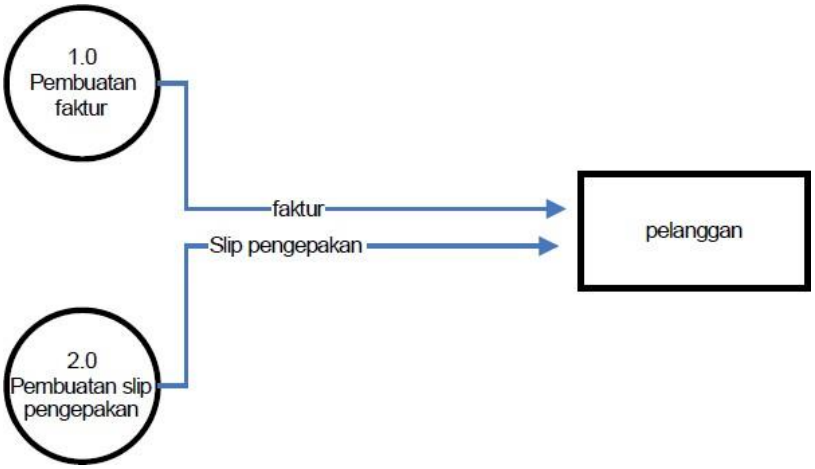
Arus data yang data yang menyebar menunjukan sejumlah tembusan dari arus data yang sama dari sumber sama ke tujuan berbeda.



Gambar 2.3 Arus Data Menyebar

3. Convergen Data Flow (Arus Data Mengumpul)

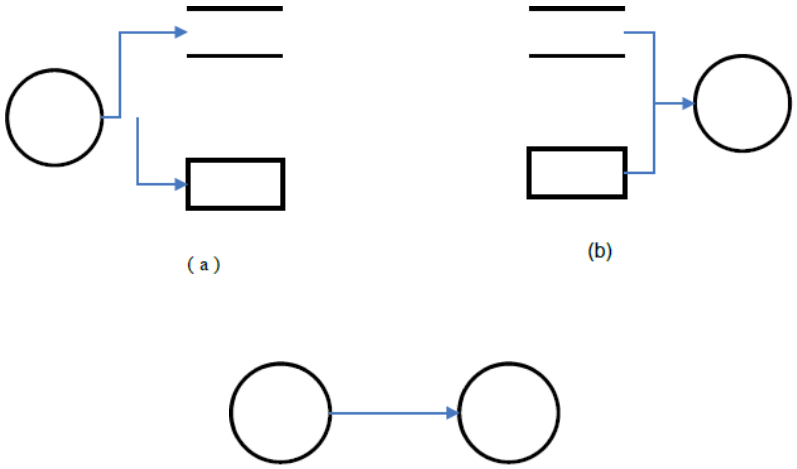
Arus data yang mengumpul, yaitu Arus data yang berbeda dari sumber yang berbeda mengumpul ke tujuan yang sama.



Gambar 2.4 Arus Data Mengumpul

4. Sumber dan Tujuan

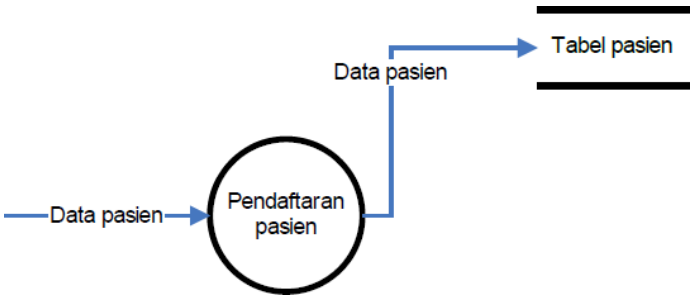
Arus data harus dihubungkan pada proses, baik dari maupun yang menuju proses.



Gambar 2.5 Sumber dan Tujuan

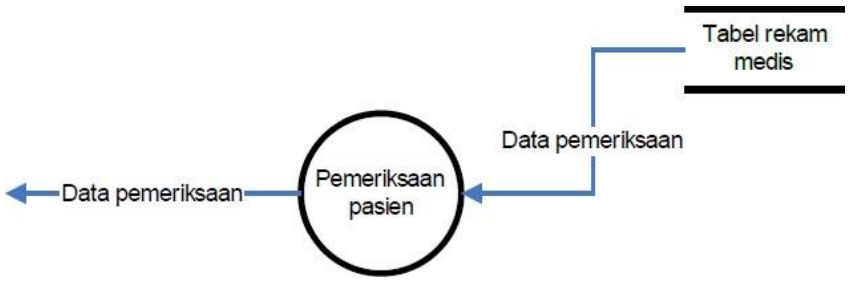
Keterangan :
a. Dari proses ke bukan proses
b. Dari bukan proses ke proses
c. Dari proses ke proses

d. ***Data Storage*** merupakan komponen untuk membuat model sekumpulan data, dapat berupa suatu file atau suatu sistem database dari suatu komputer, suatu arsip/dokumen, suatu agenda/buku.
Yang perlu diperhatikan tentang data store :
   1. Alur data dari proses menuju data store, hal ini berarti data store berfungsi sebagai tujuan/tempat penyimpanan dari suatu proses (proses write).



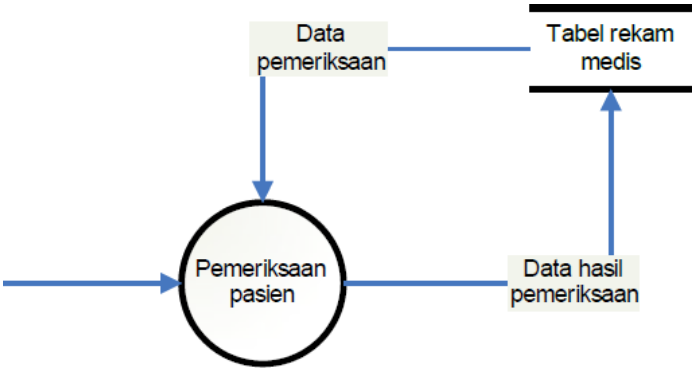
Gambar 2.6 Alur data dari proses menuju data store

   2. Alur data dari data store ke proses, hal ini berarti data store berfungsi sebagai sumber/proses yang memerlukan data (proses read)



Gambar 2.7 Alur data dari data store menuju proses

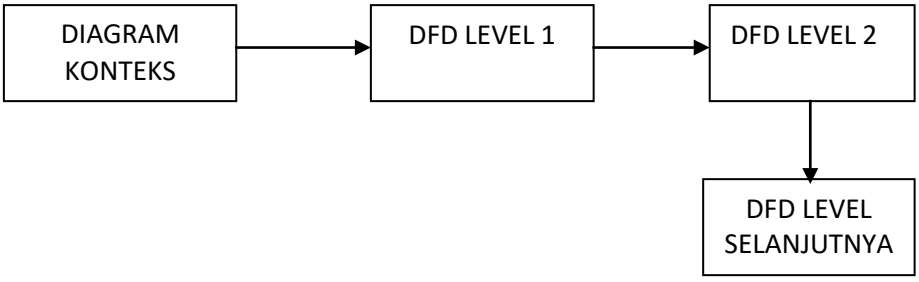
3. Alur data dari proses menuju data store dan sebaliknya berarti berfungsi sebagai sumber dan tujuan.



Gambar 2.8 Alur data dari proses menuju data store dan sebaliknya

2.3 Diagram-diagram dalam DFD

Secara umum konsep untuk menggambarkan DAD sebuah sistem dimulai dari menggambar diagram konteks, DAD Level 1, DAD Level 2 dan level selanjutnya (sesuai kebutuhan).



Gambar 2.9 diagram-diagram DFD

Ada beberapa hal yang perlu dihindari dalam pembuatan DFD, yaitu:

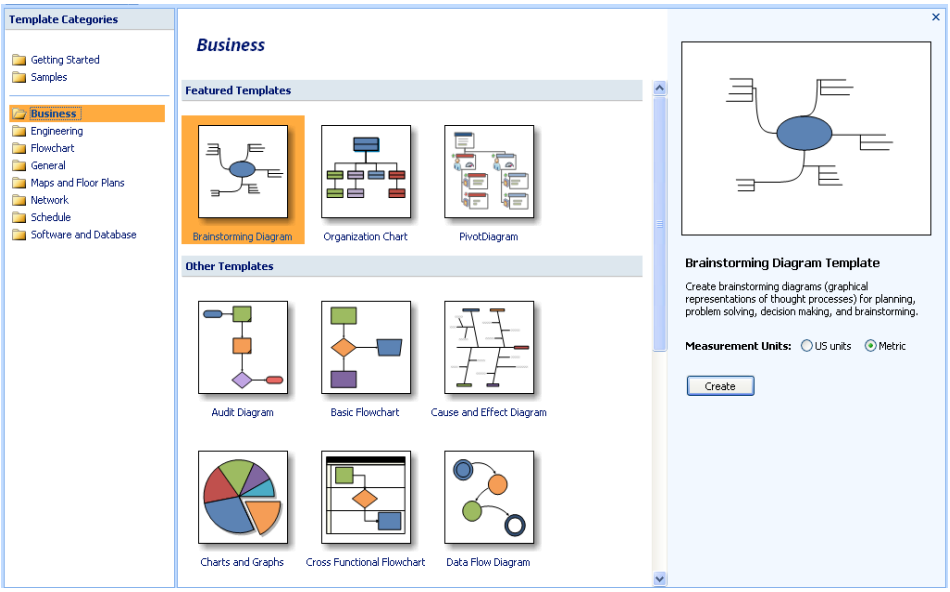
1. Arus data tidak boleh dari entitas langsung ke entitas lain tanpa melalui suatu proses.
2. Arus data tidak boleh dari data store langsung ke entitas atau sebaliknya tanpa melalui suatu proses.
3. Arus data tidak boleh dari data store langsung ke data store lainnya tanpa melalui suatu proses.
4. Sebaiknya dihindari arus data dari suatu proses langsung ke proses yang lain.

C. Praktikum

Cara untuk mengaktifkan Microsoft visio adalah sebagai berikut :
Klik menu start di windows Sorot kursor mouse di program. Pilihlah shortcut menu Microsoft Visio.
Microsoft Visio menyediakan sembilan pilihan menu utama, antara lain :

1. File : digunakan untuk mengatur yang berhubungan dengan file, seperti membuat file baru, membuka file, menutup file, mengatur file, dan lain-lain.
2. Edit : digunakan untuk proses pengeditan seperti copy, paste, delete, dan lain-lain.
3. View : digunakan untuk mengatur tampilan lingkungan kerja Visio.
4. Insert : digunakan untuk menyisipkan objek, gambar, simbol, komentar, dan lain-lain.
5. Format : digunakan untuk mengatur halaman pengerjaan.
6. Tools : digunakan untuk hal yang berhubungan dengan fasilitas tambahan yang disediakan Visio seperti Export to Database, Macro, Ruler& Grid, Report dan sebagainya.
7. Shape : digunakan untuk mengatur objek shape pada halaman pengerjaan.
8. Window : digunakan untuk mengatur jendela kerja Visio.
9. Help : digunakan untuk bantuan penggunaan Visio.

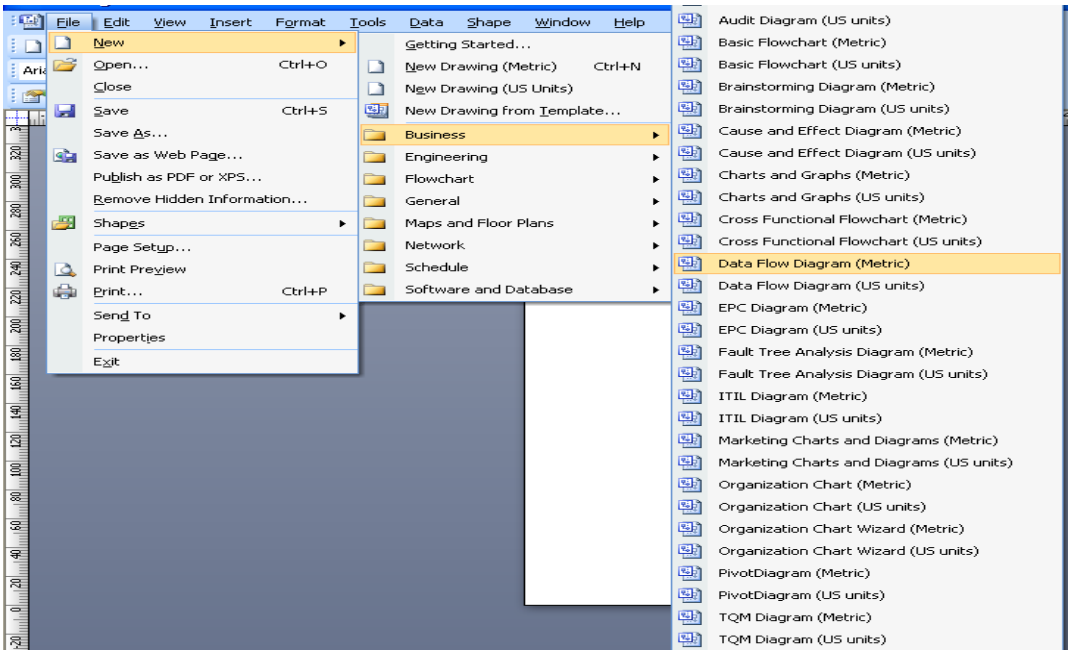
Visio menyediakan juga beberapa template untuk membantu dalam pembuatan diagram. Template dari Visio dapat dilakukan dengan memilih menu **File -> New -> Getting Started**, maka akan muncul window baru seperti terlihat pada gambar berikut:



Untuk Membuat File Baru DFD pada Microsoft Visio dapat dilakukan dengan cara :

1. Pilih menu File ->New.
2. Pilih kategori atau drawing type yang diinginkan,

Misal: Business ->Data Flow Diagram.



Langkah selanjutnya yaitu cobalah anda membuat DFD dari sistem apotek yang telah anda analisis sebelumnya di modul 1.

D. Tugas Praktikum

Tuliskan langkah-langkah pembuatan DFD yang telah anda kerjakan diatas menggunakan visio!

Modul 3
Diagram Konteks

A. Tujuan

- Mahasiswa dapat memahami level 0 dari DFD.
- Mahasiswa dapat membuat diagram konteks.

B. Dasar Teori

Diagram konteks (diagram level 0) disebut juga diagram tingkat atas, merupakan diagram sistem yang menggambarkan aliran-aliran data yang masuk dan keluar dari sistem dan yang masuk dan keluar dari entitas luar.

Hal yang harus diperhatikan :

· Memberikan gambaran tentang seluruh system.
· Terminal yang memberikan masukan ke sistem disebut *source*.
· Terminal yang menerima keluaran disebut *sink/destination*.
· Hanya ada satu proses.
· Tidak boleh ada data store.

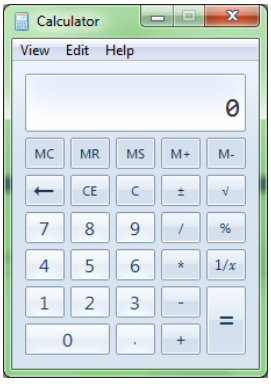
Diagram konteks merupakan diagram yang merepresentasikan seluruh elemen sistem sebagai sebuah buble tunggal dengan data input dan output yang ditunjukkan oleh anak panah yang masuk dan keluar secara berurutan, meliputi :

a. Apa saja yang dibutuhkan?
   Mencakup sistem/proses yang dipandang secara keseluruhan.
b. Siapa saja pihak yang berhubungan langsung dengan sistem (yang memberi dan menerima).
   Mencakup eksternal entiti yang terkait dengan sistem
c. Data apa yang diberikan ke sistem (input) dan yang dihasilkan (output).
   Mencakup arus data.

C. Praktikum

SKENARIO 1

Diberikan aplikasi kalkulator, seperti terlihat pada gambar 3.1. Gambarlah diagram konteks dari kalkulator.



Gambar 3.1 Aplikasi Kalkulator

Step 1 : Menentukan sistem atau proses yang dipandang secara keseluruhan dari Gambar 3.1
Nama sistem/proses → "sistem kalkulator" atau "sistem perhitungan"

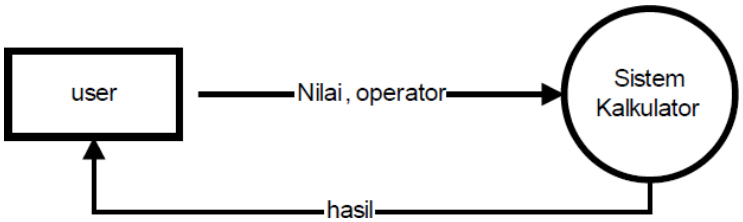
Step 2 : Menentukan pengguna sistem
Pengguna → "user"

Dalam contoh kasus ini pengguna bisa siapa saja

Step 3 : Data apa saja yang diberikan ke sistem

Data → nilai (angka), operator (fungsi perkalian, pertambahan dsb)

Step 4 : Menggambar DFD Level Konteks menggunakan microsoft viso.



Gambar 3.2 Diagram Konteks

D. Tugas Praktikum

1. Minimarket milik pak joko menjual mulai peralatan rumah tangga, alat tulis, dan barang kelontong untuk kebutuhan sehari-hari. Pak joko ingin membuat sistem yang bisa digunakan untuk transaksi penjualan di kasir. Sebagai seorang analis anda diminta untuk menggambarkan diagram konteks dari sistem kasir tersebut.
2. Pada saat ingin menggunakan facebook, pengguna diwajibkan untuk mendaftar terlebih dahulu (*sign up*). User harus menginputkan data-data pribadi untuk proses pendaftaran. Gambarkan diagram konteks dari proses pendaftaran pada facebook.

Modul 4
DFD Level 1 dan Level 2

A. Tujuan

- Mahasiswa dapat memahami tahapan levelisasi DFD.
- Mahasiswa dapat menggambar levelisasi DFD.

B. Dasar Teori

4.1 DFD Level 1

Setelah pembuatan DFD Level Konteks, selanjutnya adalah pembuatan DFD Level 1, dimana pada DFD Level adalah penggambaran dari Diagaram Konteks yang lebih rinci (Overview Diagram) atau biasanya disebut sebagai dekomposisi. Diagram ini adalah dekomposisi dari diagram Context.

Cara :

1. Tentukan proses-prose yang ada pada sistem.
2. Tentukan apa yang diberikan/diterima masing-masing proses ke/dari sistem sambil memperhatikan konsep keseimbangan (alur data yang keluar/masuk dari suatu level harus sama dengan alur data yg masuk/keluar pada level berikutnya).
3. Apabila diperlukan, munculkan data store sebagai sumber maupun tujuan alur data.
4. Gambarkan diagram level satu.
5. Hindari perpotongan arus data.
6. Beri nomor pada proses utama (nomor tidak menunjukkan urutan proses). Misalnya: 1.0, 2.0, 3.0 dst

4.2 DFD Level 2

DFD Level 2 merupakan diagram yang dibentuk dari dekomposisi proses yang terdapat pada DFD Level 1. Tidak semua proses yang terdapat pada DFD Level 1 harus di down grade (dekomposisi) ke dalam DFD Level 2, melainkan sesuai dengan kebutuhan. Jika proses yang terdapat di Diagram Level 1 butuh mencakup banyak proses di dalamnya, maka hal ini perlu dikerjakan ke dalam Diagram Level 2.

4.3 DFD Level 3 dan selanjutnya

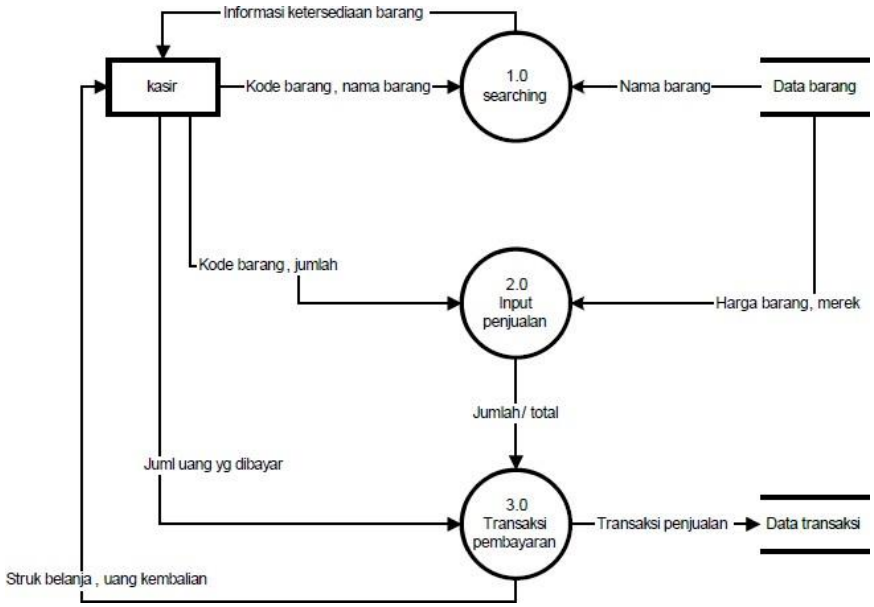
DFD Level 3 merupakan diagram yang dibentuk dari dekomposisi proses yang terdapat pada DFD Level 2.

Catatan : DFD level tiga, empat dst merupakan dekomposisi dari level sebelumnya. Proses dekomposisi dilakukan sampai dengan proses siap dituangkan ke dalam program. Aturan yang digunakan sama dengan level dua.

C. Praktikum

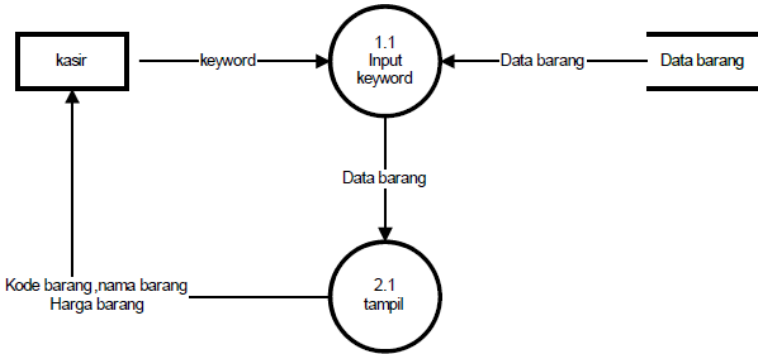
1. Minimarket milik pak joko menjual mulai peralatan rumah tangga , alat tulis, dan barang kelontong untuk kebutuhan sehari-hari. Pak joko ingin membuat sistem yang bisa digunakan untuk transaksi penjualan di kasir. Sebagai seorang analis sistem, anda diminta untuk menggambarkan DFD level 1 dari sistem kasir tersebut.

   Step 1 : Menentukan proses-proses/ event yang terjadi pada kasir
   Proses : proses pencarian barang, input transaksi, pembayaran, cetak struk.
   Step 2 : Menentukan arus data yang mengalir (input dan output) disetiap proses.
   Step 3 : Menggunakan storage untuk menyimpan data
   Step 4 : Menggambarkan ke dalam DFD

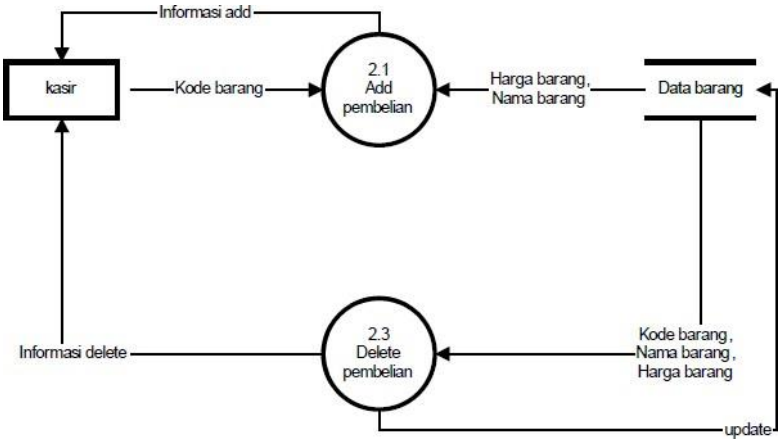


Gambar 4.1 DFD Level 1 Sistem Kasir

2. Minimarket milik pak joko menjual mulai peralatan rumah tangga, alat tulis, dan barang kelontong untuk kebutuhan sehari-hari. Pak joko ingin membuat sistem yang bisa digunakan untuk transaksi penjualan di kasir. Sebagai seorang analis sistem, anda diminta untuk menggambarkan DFD level 2 dari sistem kasir tersebut.
   Step 1 : Menentukan proses/event dari proses yang terdapat pada DFD Level 1
   Proses-proses yang terdapat pada proses searching adalah proses input keyword dan proses tampil
   Step 2 : Menentukan arus data yang mengalir (input dan output) disetiap proses.
   Step 3 : Menggunakan storage untuk menyimpan data
   Step 4 : Menggambarkan ke dalam DFD

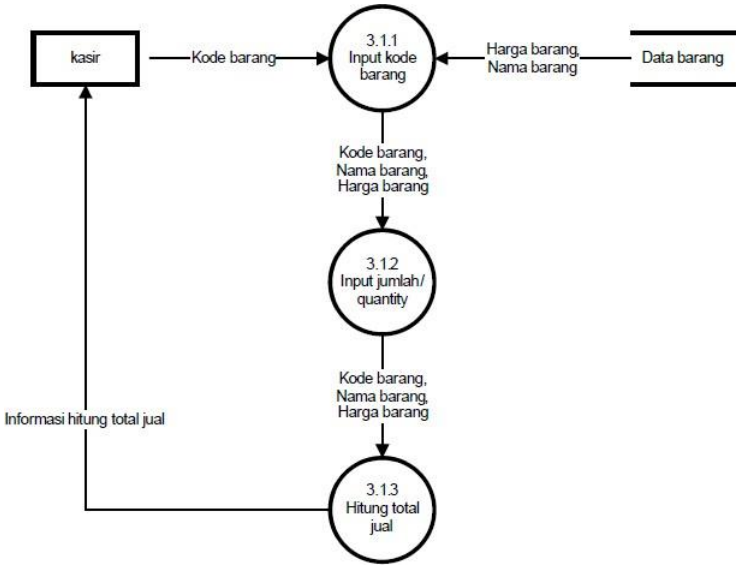


Gambar 4.2 DFD Level 2 Proses Searching



Gambar 4.3 DFD Level 2 Proses Input Penjualan

3. Langkah untuk DFD level 3.
   Step 1 :Menentukan proses/event dari proses yang terdapat pada DFD Level 2. Proses-proses yang terdapat pada proses searching adalah proses input keyword dan proses tampil
   Step 2 : Menentukan arus data yang mengalir (input dan output) disetiap proses.Step 3 : Menggunakan storage untuk menyimpan data.



Gambar 4.4 DFD Level 3 Proses Add Pembelian

D. Tugas Praktikum

Minimarket milik pak Joko menjual mulai peralatan rumah tangga , alat tulis, dan barang kelontong untuk kebutuhan sehari-hari. Pak Joko ingin membuat sistem yang bisa digunakan untuk transaksi jual beli di kasir dan manajamen data barang dagangan oleh orang kepercayaan pak Joko (admin).Selain itu sistem bisa digunakan untuk pengeloaan laporan keuangan oleh akuntan . Pak Joko sebagai direktur dapat menggunakan fitur melihat dan mencetak laporan. Sebagai seorang analis yang handal, anda bertugas untuk mengalisis kebutuhan sistem, kemudian gambarkan diagram konteks, DAD level 1,DAD level 2, dan seterusnya (jika perlu).

Modul 5
Unified Modeling Language (UML)

A. Tujuan

- Mahasiswa dapat mengetahui konsep dan bagian UML.
- Mahasiswa dapat menggambar UML menggunakan *rational rose*.

B. Dasar Teori

5.1 *Unified Modeling Language* (UML)

UML adalah bahasa untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan *artifacts* (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, artifact tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya. Selain itu UML adalah bahasa pemodelan yang menggunakan konsep orientasi object.

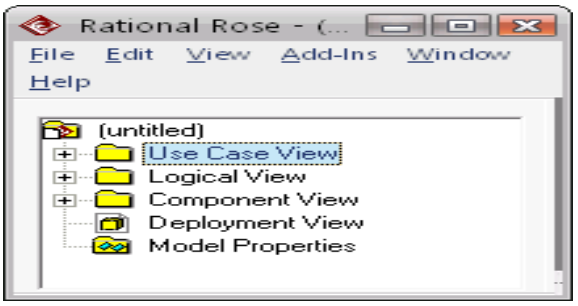
UML dibuat oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson di bawah bendera Rational Software Corp. UML menyediakan notasi-notasi yang membantu memodelkan sistem dari berbagai perspektif. UML tidak hanya digunakan dalam pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

Bagian-bagian utama dari UML adalah view, diagram, model element, dan general *mechanism*.

1. View
   View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. View bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis view dalam UML antara lain: *use case view,logical view, component view, concurrency view,* dan *deployment view*.
2. Diagram
   Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu view tertentu dan ketika digambarkan biasanya dialokasikan untuk view tertentu. Adapun jenis diagram antara lain use case diagram, class diagram, state diagram, sequence diagram, collaboration diagram, activity diagram, component diagram, deployment diagram.

5.2 Rational Rose

Rational Rose merupakan salah satu software yang paling banyak digunakan untuk melakukan design software melalui pendekatan UML (*Unified Modelling Language*). Terdapat 4 view didalam *Rational rose*, yang tiap-tiap view tersebut menjelaskan penegasan aspek yang berbeda mengenai system yang dimodelkan.



Gambar 5.1 *View dalam Rational Rose*

1. ***Use Case View*** → untuk memahami dan menggunakan system yang dimodelkan (Bagaimana actor dan use case berinteraksi). Terdapat beberapa diagram dalam view ini, yaitu :
   - *Use Case Diagram*
   - *Sequence Diagram*
   - *Collaboration Diagram*
   - *Activity Diagram*
2. ***Logical View*** → mengarah pada persyaratan fungsional sistem (kelas-kelas dan hubungan antar kelas tersebut). Terdapat beberapa diagram dalam view ini, yaitu:
   - *Class Diagram*
   - *Sequence Diagram*
   - *Collaboration Diagram*
   - *Statechart Diagram*
3. ***Componen View*** → Pengaturan software (informasi komponen software, komponen tereksekusi dan library untuk system yang dimodelkan). Satu jenis diagram yang terdapat view ini, yaitu Componen Diagram.
4. ***Deployment View*** → Pemetaan setiap proses kedalam hardware. Satu jenis diagram yang terdapat pada view ini, yaitu Deployment Diagram.

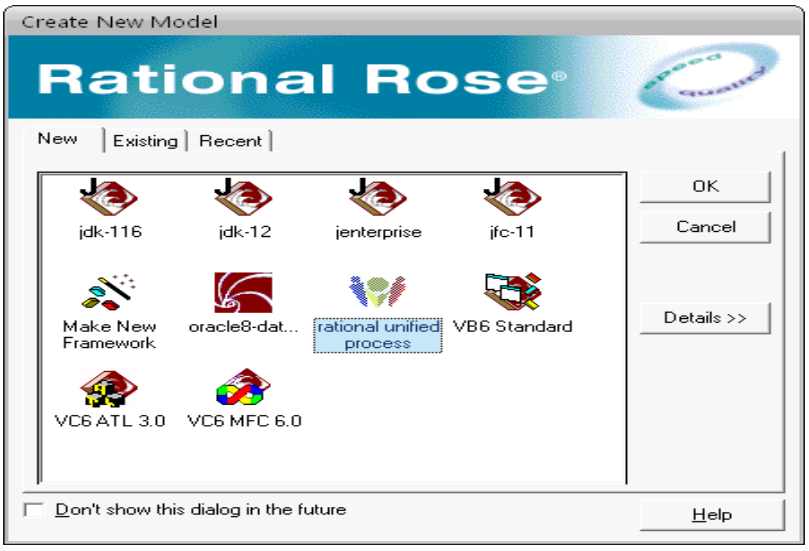
Tujuan Penggunaan UML yaitu:

1. Memodelkan suatu sistem (bukan hanya perangkat lunak) yang menggunakan konsep berorientasi object.
2. Menciptakan suatu bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin.

C. Praktikum

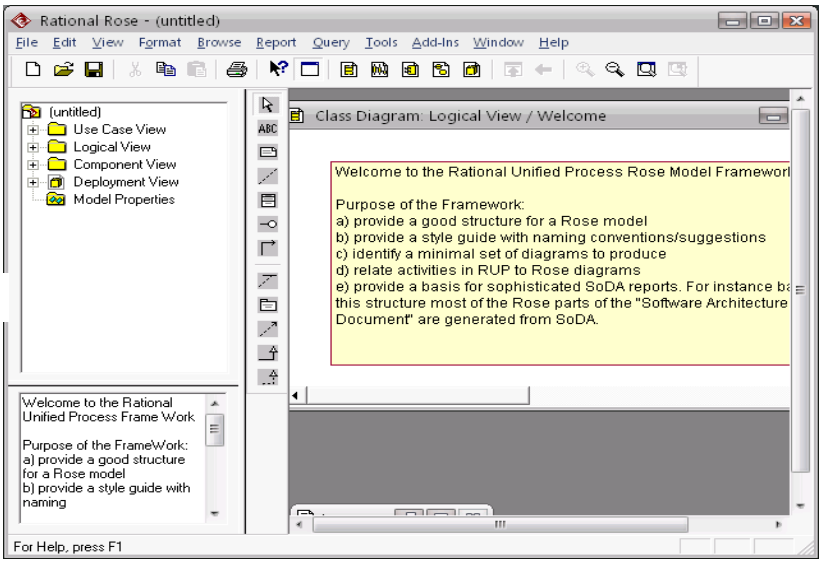
Menjalankan Rational Rose:

1. Jalankan Rational Rose dari **Start Menu – Programs – Rational Rose 2000 Enterprise Edition – Rational Rose 2000 Enterprise Edition**.
2. Pada jendela **Create New Model** pada tab **New** terdapat ikon–ikon yang merupakan framework yang telah disediakan bagi pemakai Rose. Ini merupakan pilihan apakah ingin membuat file yang mengandung komponen Java, VB, atau VC++.



Gambar 5.2 *Rational Rose*

3. Pilihlah ***Rational Unified Proses*** (merupakan template untuk membuat design yang lengkap). Kemudian akan muncul Diagram yang bernama Welcome pada Logical View.



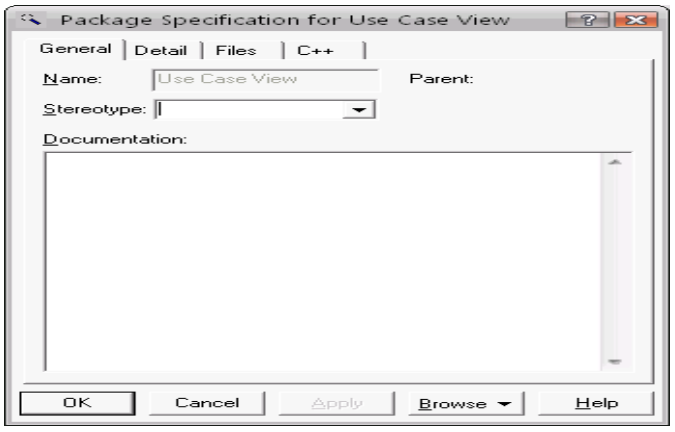
Browser

JendelaDokumentasi

Gambar 5.3 *Diagram Welcome pada Rational Rose*

- Jendela Diagram → Membuat, menampilkan atau memodifikasi diagram.
- Jendela Dokumentasi → Membuat Dokumentasi elemen-elemen model.
- Browser → Melihat secara hirarkis elemen-elemen model.
- Toolbox → Toolbox in berubah sesuai jenis diagram yang aktif.

4. Klik Kanan Use case view pada browser, kemudian pilih ***Open Spesification***. Sehingga akan muncul jendela berikut:



Gambar 5.4 *Jendela Open Spesification*

- JendelaSpesification → Membuat atau mengubah properties elemen model.
- Stereotype → Menerangkan subklasifikasi dari sebuah elemen model.
- Package → Mekanisme pengelompokan elemen-elemen model, untuk mempermudah pengorganisasiannya.

D. Tugas Praktikum

1. Buatlah contoh keadaan nyata dengan menggunakan pendekatan bagian-bagian dari UML view.
2. Di dalam permasalahan sistem yang komplek, apakah pemodelan UML dapat digunakan dengan baik? Jelaskan jawaban anda!

Modul 6
Use Case Diagram

A. Tujuan

- Mahasiswa mampu membuat sebuah skenario suatu sistem yang nantinya dapat diimplementasikan menjadi sebuah perangkat lunak.
- Mahasiswa bisa memahami alur dari setiap tahap yang digunakan dalam perancangan perangkat lunak menggunakan UML.
- Mahasiswa dapat memahami hubungan atara actor dengan use case diagram.
- Mahasiswa mampu membuat use case diagram dari skenario yang telah ada.

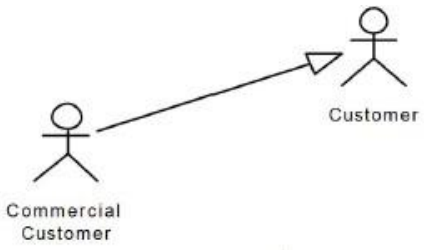
B. Dasar Teori

Use case diagram digunakan untuk menggambarkan sejumlah external actors dan hubungannya ke use case yang diberikan oleh sistem. Use case adalah deskripsi fungsi yang disediakan oleh system dalam bentuk teks sebagai dokumentasi dari use casesymbol namun dapat juga dilakukan dalam activity diagram. Use case digambarkan hanya yang dilihat dari luar oleh actor (keadaan lingkungan sistem yang dilihat user) dan bukan bagaimana fungsi yang ada di dalam sistem.

a. Actor

Pada dasarnya actor bukanlah bagian dari use case diagram, namun untuk dapat terciptanya suatu use case diagram diperlukan beberapa actor dimana actor tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah actor mungkin hanya memberikan informasi inputan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima dan memberi informasi pada sistem, actor hanya berinteraksi dengan use case tetapi tidak memiliki kontrol atas use case. Actor digambarkan dengan stick man. Actor dapat digambarkan secara secara umum atau spesifik, dimana untuk membedakannya kita dapat menggunakan relationship.

Contoh :



Gambar 6.1 Actor

Ada beberapa kemungkinan yang menyebabkan actor tersebut terkait dengan sistem antara lain:

- Yang berkepentingan terhadap sistem dimana adanya arus informasi baik yang diterimanya maupun yang dia inputkan ke sistem.
- Orang ataupun pihak yang akan mengelola sistem tersebut.
- External resource yang digunakan oleh sistem.
- Sistem lain yang berinteraksi dengan sistem yang akan dibuat.

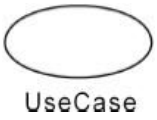
b. Use Case

Use case adalah gambaran fungsionalitas dari suatu sistem, sehingga customer atau

pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun.

Catatan:

Use case diagram adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (user), sehingga pembuatan use case lebih dititik beratkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian.



Gambar 6..2 Use Case

c. Relasi dalam Use Case

Ada beberapa relasi yang terdapat pada use case diagram:

1. ***Association***, menghubungkan link antar element.
2. ***Generalization***, disebut juga inheritance (pewarisan), sebuah elemen dapat merupakan spesialisasi dari elemen lainnya.
3. ***Dependency***, sebuah element bergantung dalam beberapa cara ke element lainnya.
4. ***Aggregation***, bentuk assosiation dimana sebuah elemen berisi elemen lainnya.

Tipe relasi/ stereotype yang mungkin terjadi pada use case diagram:

1. **<<include>>**, yaitu kelakuan yang harus terpenuhi agar sebuah event dapat terjadi, dimana pada kondisi ini sebuah use case adalah bagian dari use case lainnya.
2. **<<extends>>**, kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakkan alarm.
3. **<<communicates>>,** mungkin ditambahkan untuk asosiasi yang menunjukkan asosiasinya adalah communicates association. Ini merupakan pilihan selama asosiasi hanya tipe ralationship yang dibolehkan antara actor dan use case.

C. Praktikum

SKENARIO:

Sebuah bank mengoperasikan ATM dan mengelola banyak tabungan, setiap nasabah memiliki setidaknya satu rekening tabungan pada satu bank tertentu. Setiap tabungan dapat diakses melalui kartu debit. Proses utama sistem ATM berkomunikasi dengan pusat komputer dan didesain untuk menangani beberapa transaksi. Setiap transaksi menunjuk sebuah tabungan tertentu. Suatu transaksi akan menghasilkan satu dari dua hal berikut: transaksi diterima atau mengeluarkan pesan penolakan transaksi".

Untuk melakukan sebuah transaksi akan melalui dua tahap: pengecekan tabungan dan pemroses transaksi. Proses pengecekan tabungan akan menetapkan persetujuan untuk proses transaksi. Jika persetujuan ditolak, ATM akan mengeluarkan pesan penolakan, namun jika diterima, transaksi akan diproses de ngan menggunakan nomor rekening tabungan dan ATM membaca dari kartu debit.

Pengecekan tabungan dilakukan bersamaan pada saat ATM memvalidasi kartu debit dari bank yang bersangkutan. Jika kartu valid, password akan dicek dengan nasabah.

Untuk memudahkan kita dalam menganalisa skenario yang akan kita gunakan pada fase- fase selanjutnya maka kita dapat melakukan pemilahan terhadap skenario tersebut, antara lain:

Skenario use case

Nama use case : Authenticate user
Actor : User, bank
Type : Primary
Tujuan : verifikasi user

Tabel 6.5.2.1 Skenario Authenticate User

ACTOR	SISTEM
1. User memasukkan kartu debit	
	2. ATM meminta PIN dari user
3. User memasukkan PIN dan menekan OK	
	4. ATM memverifikasi dengan Bank bahwa kartu dan PIN adalah legal dari Rekening yang benar
	5. ATM meminta jenis transaksi

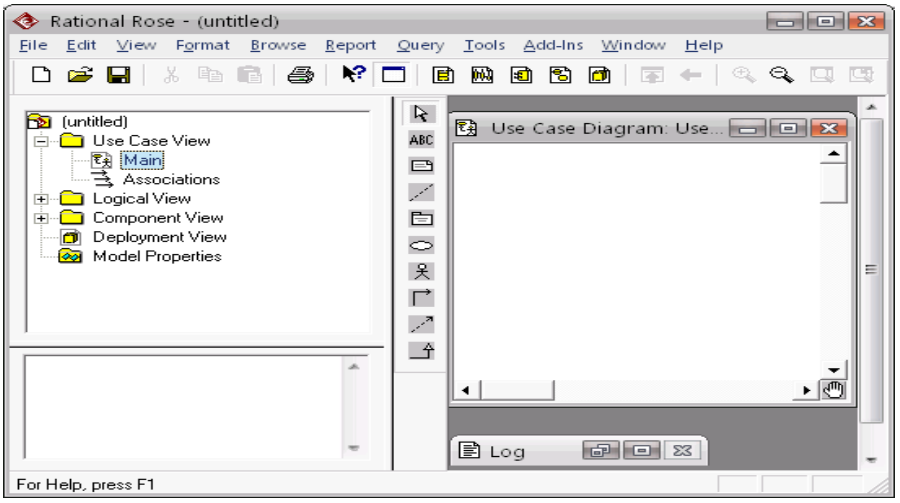
Nama use case : Withdrawal
Actors : User, bank
Type : Primary
Tujuan : Penarikan uang secara cash
Deskripsi : User datang ke ATM dengan kartu debit untuk melakukan penarikan tunai. User memasukkan kartu ke ATM. ATM meminta user untuk memasukkan PIN. User memasukkan PIN dan sistem mengotorisasi penarikan tunai. ATM mengeluarkan uang dan mengeluarkan nota. ATM mengirim transaction record ke bank untuk meng-update saldo tabungan. Setelah selesai, user meninggalkan ATM dengan membawa uang dan nota tadi.

Tabel 6.5.2.2 Skenario Withdrawal

ACTOR	SISTEM
1. User memilih menu withdrawal	
	2. ATM meminta jumlah uang yang akan ditarik
3. User memasukkan jumlah uang yang akan ditarik	
	4. ATM mencek jumlah uang yang akan ditarik dengan saldo minimal yang diperbolehkan pada bank tersebut.
	5. Update saldo
	6. ATM mengeluarkan uang
	7. ATM mencetak nota dan mengeluarkan kartu

Untuk membuat diagram use case dalam Rational Rose, ikuti langkah berikut:

1. Klik tanda + di sebelah kiri *Use Case View* pada browser.
2. Klik 2 kali Main di bawah *Use Case View*, sehingga muncul jendela *Use Case Diagram* seperti berikut :



Cobalah membuat *use case diagram* dari skenario diatas!

D. Tugas Praktikum

1. Buatlah skenario use case dari sebuah sistem informasi perpustakaan!
2. Buatlah use case diagram dari sistem informasi perpustakann tersebut dan jelaskan langkah-langkah pembuatannya menggunakan rasional rose!

Modul 7
Class Diagram

A. Tujuan

- Mahasiswa dapat menggambarkan class diagram dari candidate class yang telah ada.
- Mahasiswa bisa memberikan atribut dan operasi pada masing-masing class yang didefenisikan.
- Mahasiswa dapat menggambarkan relasi antar class dan tipe - tipenya.

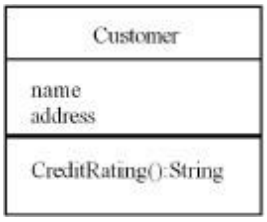
B. Dasar Teori

7.1 Definisi class dan object

Class diagram menggambarkan struktur dan deskripsi class, package, dan objek berserta hubungan antar-class di dalam sistem seperti containment, pewarisan, asosiasi dan lain-lain.

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

Class digambarkan dengan sebuah kotak dibagi menjadi tiga bagian. Bagian paling atas diisikan nama class, bagian tengah diiisikan variabel/atribut yang dimiliki class, dan bagian bawah diisikan method-method dari class.



Gambar 7.1 Class

7.2 Status (state), behaviour dan identity

Status dari object adalah satu kondisi yang mungkin ada. Status dari object akan berubah setiap waktu dan ditentukan oleh sejumlah property (atribut) dengan nilai dari properti, ditambah relasi object dengan object lainnya.

Sifat (Behaviour) menentukan bagaimana object merespon permintaan dari object lain dan melambangkan setiap object yang dapat dilakukan. Sifat ini diimplementasikan dengan sejumlah operasi untuk object.

Identitas (Identify) artinya setiap object yang unik Pada UML, object digambarkan dengan segiempat dan nama dari object diberi garis bawah.

7.3 Atribut

Attribut adalah salah satu property yang dimiliki oleh class yang menggambarkan batasan dari nilai yang dapat dimiliki oleh property tersebut. Sebuah class mungkin memiliki beberapa atribut atau tidak memilikinya sama sekali. Sebuah atribut merepresentasikan beberapa property dari sesuatu yang kita modelkan, yang dibagi dengan semua object dari semua class yang ada. Contohnya, setiap tembok memiliki tinggi, lebar dan ketebalan. Atribut dalam implementasinya akan digambarkan sebagai

sebuah daftar (list) yang diletakkan pada kotak dibawah nama class. Ia seperti halnya nama class merupakan teks. Biasanya huruf pertama dari tiap kata merupakan huruf kapital, terkecuali untuk huruf awal. Sebagai contohnya : birthDate.

Gambar 7.2 Atribut dari Class Diagram

Untuk lebih lanjut kita pun bisa menspesifikasikan atribut beserta jenis data yang kita gunakan untuk atribut tersebut.

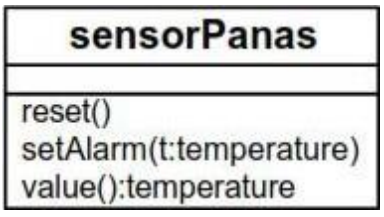
Gambar 7.3 Atribut dan jenis data dari Class Diagram

7.4 Operasi

Sebuah operasi adalah sebuah implementasi dari layanan yang dapat diminta dari beberapa object dari class , yang mempengaruhi behaviour. Dengan kata lain operasi adalah abstraksi dari segala sesuatu yang dapat kita lakukan pada sebuah object dan ia berlaku untuk semua object yang terdapat dalam class tersebut. Class mungkin memiliki beberapa operasi atau tanpa operasi sama sekali.contohnya adalah sebuah class “kotak” dapat dipindahkan, diperbesar atau diperkecil. Biasanya (namun tidak selalu), memanggil operasi pada sebuah object akan mengubah data atau kondisi dari object tersebut. Operasi ini dalam implementasinya digambarkan dibawah atribut dari sebuah class.

Gambar 7.4 Contoh dari Operasi

Untuk lebih lanjut kita pun bisa menspesifikasikan semua parameter yang terlibatdalam operasi tersebut.



Gambar 7.5 Contoh lain dari Operasi

7.5 Pengorganisasian Atribut dan Operasi

Ketika menggambarkan sebuah class kita tidak perlu menampilkan seluruh atribut atau operasi. Karena dalam sebagian besar kasus kita tidak dapat menampilkannya dalam sebuah gambar, karena telalu banyaknya atribut atau operasinya bahkan terkadang tidak perlu karena kurang relevannya atribut atau operasi tersebut untuk ditampilkan. Sehingga kita dapat menampilkan hanya sebagian atau bahkan tidak sama sekali atribut dan operasinya.

Kosongnya kotak tempat pengisian bukan berarti tidak ada. Karena itu kita dapat menambahkan tanda ("…") pada akhir daftar yang menunjukkan bahwa masih ada atribut atau operasi yang lain.

Atribut dan method dapat memiliki salah satu dari sifat berikut:

1. **private**, tidak dapat dipanggil dari luar class yang bersangkutan
2. **protected,** hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya.
3. **public**, dapat dipanggil oleh siapa saja

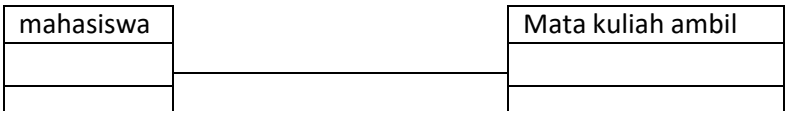
7.6 Relasi pada class diagram

a. Generalization dan Inheritence

Hubungan antara class yang bersifat umum (superclass atau parent class) dengan class yang bersifat lebih spesifik (subclass atau child class). Sebuah generalization dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas "parent"nya/induknya.

b. Associations

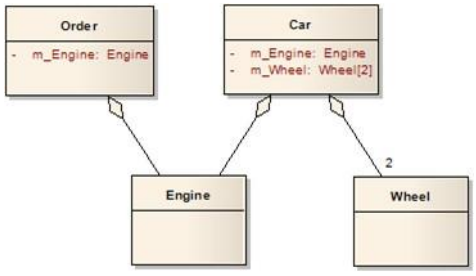
Association adalah hubungan struktural antar class. Ini bukan aliran data sebagaimana pada pemodelan desain dan analisa terstruktur, data diperbolehkan mengalir dari kedua arah. Asosiasi diantara class-class artinya ada hubungan antara object-object pada class-class yang berhubungan. Banyaknya object yang terhubung tergantung dengan *multiplicity* pada asosiasi, yang akan dibahas nanti. Relasi asosiasi umumnya menggambarkan class yang memiliki atribt berupa class lain, atau class yang harus mengetahui ekstensi class lain. Dalam notasi uml kita mengenal asosiasi 2 arah (bidirectional) dan 1 arah (undirectional).



Gambar 7.6 Contoh associations

c. Aggregation

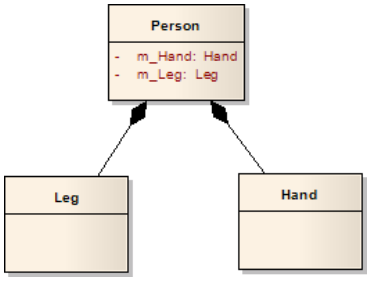
Hubungan antar-class dimana class yang satu (part class) adalah bagian dari class lainnya (whole class). Digambarkan dengan garis dan belah ketupat diujung.



Gambar 7.7 Contoh aggregation

d. Composition

Aggregation dengan ikatan yang lebih kuat. Didalam composition aggregation, siklus hidup part class sangat bergantung pada whole class sehingga bila objek instance dari whole class dihapus maka objek instance dari part class juga akan terhapus.

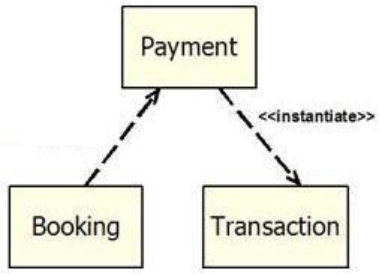


Gambar 7.8 Contoh composition

e. Dependency

Hubungan antar-class dimana sebuah class memiliki ketergantungan pada class lainnya tetapi tidak sebaliknya.

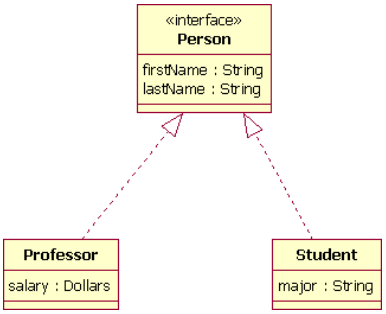
Contoh class peminjaman, jika ada relasi dependency dari class peminjam ke class denda, maka class denda akan bergantung ke class peminjam tetapi tidak sebaliknya.



Gambar 7.9 Contoh dependency

f. Realization

Hubungan antar class dimana sebuah class memiliki keharusan untuk mengikuti aturan yang ditetapkan class lainnya. Biasanya realization digunakan untuk menspesifikasikan hubungan antara sebuah interface dengan class yang mengimplementasikan interface tersebut.



Gambar 7.10 Contoh realization

7.7 Multiplisitas Relasi

Multiplicity atau ultiplisitas adalah jumlah banyaknya objek sebuah class yang berelasi dengan sebuah objek lain pada class lain yang berasosiasi dengan class tersebut. Untuk menyatakan multiplisitas anda dapat meletakkannya diatas garis

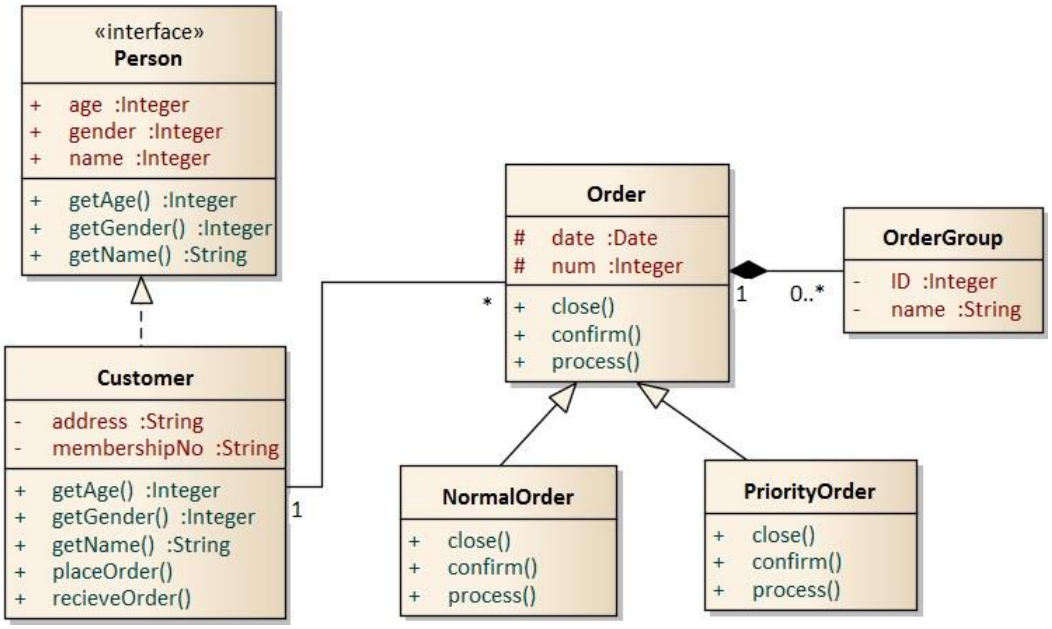
asosiasi berdekatan dengan class yang sesuai. Ada banyak multiplisitas yang mungkin untuk dipakai. Tabel berikut menjabarkan multiplisitas yang dapat digunakan.

Tabel 7.1 Multiplisitas

Multiplisitas	Arti
*	Banyak
0	Nol
1	Satu
0...*	Nol atau banyak
1...*	Satu atau banyak
0...1	Nol atau satu
1...1	Hanya satu

C. Praktikum

Buatlah class diagram dibawah ini menggunakan rational rose!



Gambar 7.11 Class Diagram

D. Tugas Praktikum

Buatlah sebuah class diagram dari sistem informasi perpustakaan!

Modul 8
Sequnce Diagram

A. Tujuan

- Mahasiswa memahami konsep sequnce diagram.
- Mahasiswa dapat membuat sequence diagram dan mengimplementasikan sequence diagram.

B. Dasar Teori

Sequence diagram merupakan gambaran interaksi antar objek dalam dan disekitar sistem (termasuk pengguna dan display) yang saling berkomunikasi menggunakan pesan (message) dan memiliki parameter waktu.

Fungsi utama sequence diagram yaitu untuk menggambarkan skenario atau urutan langkah-langkah yang dilakukan sebagai respon yang memicu aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Sequence diagram bersifat inter-class berbeda dengan activity diagram yang bersifat inner-class. Sequence diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

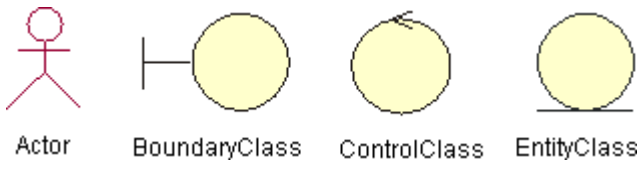
Di dalam diagram sequence, setiap objek hanya memiliki satu garis yang digambarkan garis putus-putus ke bawah (lifeline). Pesan antar objek digambarkan dengan anak panah dari objek yang mengirim pesan ke objek penerima pesan ke objek yang menerima pesan. Dan sebaiknya setiap sebuah usecase dibuat satu sequence diagram.

Kegunaan sequnce diagram adalah untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antar objek yang terjadi pada titik tertentu dalam eksekusi suatu sistem.

Di dalam Sequence Diagram, terdapat pelaku (actor), boundary class, control class, dan entity class.

- ***Boundary Class*** adalah kelas yang memodelkan interaksi antara satu atau lebih actor dengan system. Boundary memodelkan bagian dari system yang bergantung pada pihak lain disekitarnya dan merupakan pembatas system dengan dunia luar.
- ***Control Class*** digunakan untuk memodelkan “perilaku mengatur”, khusus untuk satu atau beberapa use-case saja.
- ***Entity Class*** memodelkan informasi yang harus disimpan oleh system. Entity Class memperlihatkan struktur data dari suatu system.

Adapun notasi dari actor, boundary class, control class dan entity class adalah sebagai berikut :



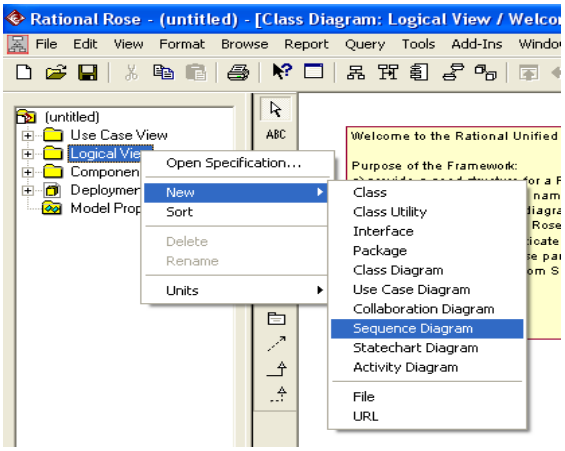
Gambar 8.1 Komponen Sequence Diagram

Aturan pembuatan diagram sequnce:

1. Setiap objek, termasuk aktor memiliki life-line vertika yang dilambangkan dengan garis putus-putus.
2. Message digambarkan dengan garis berpanah (→) dari satu objek ke objek lain. Pada fase desain berikutnya, message akan dipetakan menjadi operasi method dari class.
3. Aktivasi bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.
4. Notasi objek secara umum dilambangkan dengan persegi panjang.

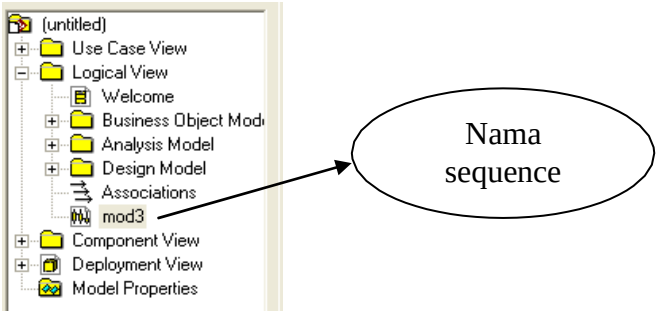
C. Praktikum

Dari menu yang terdapat di sebelah kanan, terdapat tulisan Logical View, pilih New, klik Sequency Diagram.



Gambar 8.2 Menu Sequence Diagram

Setelah muncul nama, maka ganti dengan nama mod3, seperti pada gambar berikut ini :



Gambar 8.3 Nama sequence

1. Gambarkan actor dan kelas yang terlibat ke dalam sequence diagram.
2. Urutkan sebagai berikut : Actor – obyek dari boundary class – obyek dari control class- obyek entity class.
3. Ubah dari tipe kelas analisa menjadi kelas desain.
4. Ikuti urutan seperti dalam use case spesification dan mulai identifikasi operasi yang diperlukan untuk mengeksekusi suatu baris aktivitas dalam use case spesification.
5. Dari masing-masing operasi tersebut, identifikasi informasi apa saja yang perlu dipindahkan dari actor ke boundary class ke control class hingga ke entity class dan informasi apa yang harus dikembalikan dari entity class ke boundary class, control class atau ke actor. Untuk satu use case bisa dibuat beberapa sequence diagram, karena satu use case biasanya terdiri dari beberapa aktivitas yang dilakukan dan masing-masing aktivitas ini bisa direpresentasikan dalam satu sequency diagram.

6. Tombol-tombol yang terdapat dalam sequence diagram adalah sebagai berikut :

Gambar 8.4 Tombol sequence diagram

Praktekkan cara membuat Sequence Diagram seperti petunjuk di atas untuk membuat sequence diagram dari class diagram yang ada di modul 7!

D. Tugas Praktikum

Buatlah sequence diagram dari sistem informasi perpustakaan! (minimal 2 sequnce diagram).

Modul 9
Activity Diagram

A. Tujuan

- Mahasiswa dapat memhami konsep activity diagram.
- Mahasiswa dapat membuat activity diagram.

B. Dasar Teori

9.1 Activity Diagram

Activity diagram berdasarkan asal katanya dapat didefinisikan sebagai diagram yang menggambarkan sebuah aktivitas project atau sistem yang akan dibangun. Aktivitas diagram akan menggambarkan berbagai aliran aktifitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, decision yang mungkin terjadi, dan bagaimana aktivitas itu berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Pada umumnya activity diagram tidak menampilkan secara detail urutan proses, namun hanya memberikan gambaran global bagaimana urutan prosesnya. Sehingga seringkali diagram ini digunakan untuk memodelkan aktivitas bisnis dalam level konseptual. Diagram ini mirip dengan flowchart karena kita dapat memodelkan sebuah alur kerja dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas ke dalam sesaat (state). Akan tetapi perbedaannya dengan flowchart adalah activity diagram dapat mendukung perilaku pararel sedangkan flowchart tidak.

Membuat activity diagram terlebih dahulu dalam memodelkan sebuah proses dapat membantu kita memahami proses secara keseluruhan. Activity diagram juga berguna ketika kita ingin menjelaskan bagaimana perilaku dalam berbagai use case berinteraksi.

Sebuah aktivitas dapat direalisasikan oleh satu use case atau lebih. Aktivitas menggambarkan proses yang berjalan sementara use case menggambarkan bagaimana actor menggunakan sistem untuk melakukan aktivitas. Activity diagram adalah variasi dari state diagram yang mana “state” merepresentasikan operasi, dan transisinya merepresentasikan aktivitas yang terjadi pada saat operasi sudah selesai. Sama seperti state, standar UML menggunakan segi empat dengan sudut membulat untuk menggambarkan aktivitas.

Decision digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (fork and joint) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal, atau vertical. Activity diagram dapat dibagi menjadi beberapa object swimlane untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

Activity diagram digunakan untuk menggambarkan semua aktifitas secara global yang terjadi dalam sebuah sistem.

Dengan melihat activity diagram, pengguna dapat mengetahui apa saja yang dapat dilakukan pada sebuah sistem. Selain itu, activity diagram juga bermanfaat untuk menggambarkan pararel behaviour atau menggambarkan interaksi antara beberapa use case.

9.2 Komponen Activity Diagram

1. ***Start State***
   Start state adalah komponen activity diagram untuk mengawali aliran kerja (*workflow*). Start state menggambarkan awal dari sebuah aktifitas. Gambar buttonnya:

2. ***End State***
   End State adalah komponen activity diagram untuk mengakhiri aliran kerja (*workflow*). Start state menggambarkan akhir dari sebuah aktifitas. Gambar buttonnya:

3. ***Activity State***
   Activity state adalah komponen activity diagram untuk menggambarkan sebuah state atau aktifitas yang masih dapat dipecah lagi atau diturunkan lagi menjadi sebuah aktivitas yang lebih kecil-kecil lagi hingga akhirnya aktivitas tersebut tidak bisa diturunkan kembali. Activity state biasa disebut dengan state saja.
   Sebagai contoh yaitu aktivitas menggunakan motor. Jika akan menggunakan motor, langkah apa yang kita lakukan? Pertama kita masukkan kunci motor, kedua nyalakan motor. Nah, aktivitas kedua tersebut merupakan sebuah activity state, karena aktivitas menyalakan motor tersebut masih bisa kita turunkan lagi, misalnya cara meyalakan motor tersebut pertama kita tekan starter, kemudian tarik gas, dan seterusnya. Gambar buttonnya:

4. ***Action State***
   Action state adalah komponen activity diagram yang merupakan aktivitas turunan dari Activity state. Jika activity state adalah sebuah aktivitas yang masih bisa dipecah lagi, sedangkan action state merupakan kebalikannya. Kalau menurut contoh diatas, yang merupakan action state adalah “masukkan kunci motor”, karena aktivitas masukkan kunci motor tidak bisa diturunkan lagi. Gambar buttonnya:

5. ***Transition***
   Transition adalah komponen activity diagram untuk menghubungkan antara satu action/activity dengan action/activity lainnya. Ketika aksi atau activity dari suatustate diselesaikan, aliran kendali akan menuju ke aksi atau aktivitas berikutnya. Gambar buttonnya:

6. ***Forking dan joining***
   Forking adalah satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran. Sedangkan joining adalah beberapa aliran sekaligus yang secara bersamaan masuk menjadi satu titik. Pada rational rose, forking dan joining digambarkan dengan komponen button synchronization. Synchronization itu sendiri terbagi menjadi dua, yaitu:
   a. *Horizontal Synchronization*, yaitu menambah sinkronisasi horizontal pada diagram.

b. *Vertical Synchronization*, yaitu menambah sinkronisasi vertical pada diagram.

7. Decision

Decision adalah komponen untuk menambah titik keputusan pada aliran kerja. Komponen ini berhubungan dengan branching (percabangan) yang telah dijelaskan diatas. Gambar buttonnya:

8. Swimlane

Swimlane adalah komponen untuk memperlihatkan siapa yang bertanggung jawab untuk melaksanakan tugas-tugas tertentu pada activity diagram. Swimlane ini umumnya digunakan pada pemodelan bisnis. Gambar buttonnya:

9. Note

Note adalah komponen untuk menambahkan catatan pada diagram. Gambar buttonnya:

10. Text box

Text box adalah komponen untuk menambahkan box text ke diagram. Gambar buttonnya:

11. Select

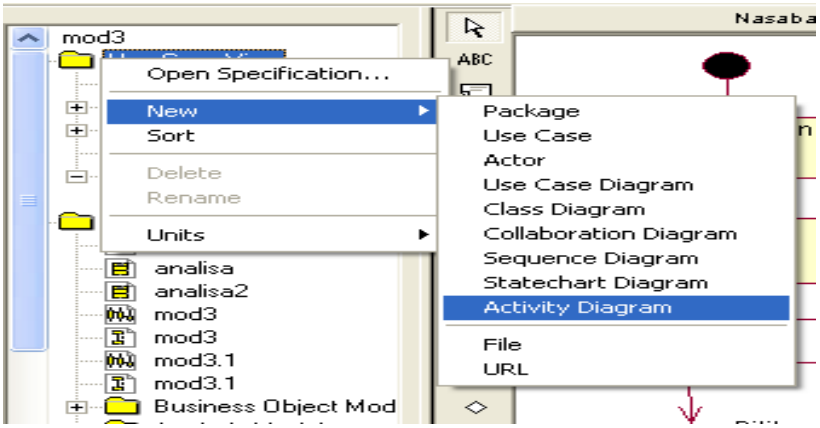
Select adalah komponen untuk mengembalikan pointer mouse ke icon select memungkinkan kita untuk memilih icon yang lainnya. Gambar buttonnya:

12. Anchor Note to item

Komponen untuk melekatkan catatan pada state atau activity tertentu dalam diagram. Gambar buttonnya:

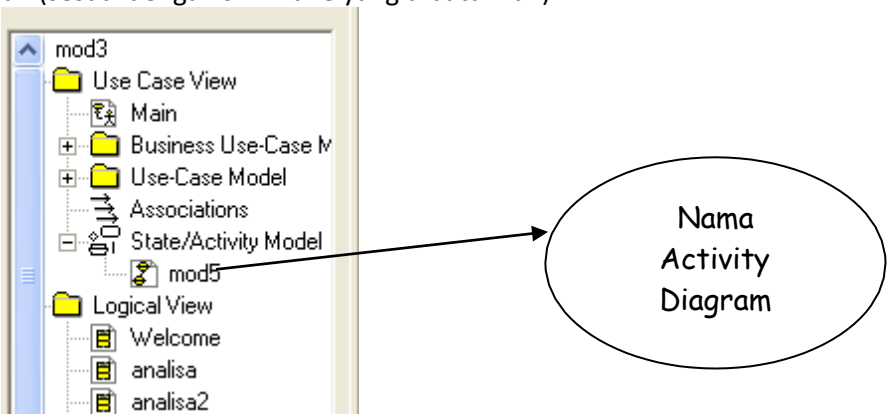
C. Praktikum

1. Untuk memulai klik Dari menu Start, klikPrograms, pilih Programming, pilih Rational Rose 2000 Enterprise Edition.
2. Klik Rational Unified Process dari menu yang tampil, lalu klik OK.
3. Dari menu yang terdapat di sebelah kanan, terdapat tulisan Use Case View, pilih New, klik Activity Diagram.
4. Setelah muncul nama, maka ganti dengan nama mod5, seperti pada gambar berikut ini :



Gambar 9.1 Menu Activity Diagram

5. Dari menu Tools, pilih Create, pilih Swimlane, kemudian drag di layar sesuai dengan kebutuhan (sesuai dengan swimlane yang dibutuhkan).



Gambar 9.2 Nama Activity Diagram

Buatlah activity diagram dari studi kasus yang ada di modul 6!

D. Tugas Praktikum

Buatlah acitivity diagram dari sistem informasi perpustakaan! (minimal 3 activity diagram).