MODUL 3

UML DENGAN RATIONAL ROSE

Daftar Isi

3.1. Pengenalan Dasar Rational Rose

3.1.1. Apakah Rational Rose Itu?

Rational Rose adalah alat pemodelan visual yang digunakan untuk membantu dalam analisis dan desain sistem perangkat lunak berorientasi obyek.

Rational Rose digunakan untuk memodelkan sistem sebelum coding dibuat, sehingga kita dapat memastikan bahwa sistem tersebut secara arsitektur sudah baik dari awalnya. Dengan menggunakan sebuah model kita dapat menemukan kelemahan lebih awal sehingga biaya untuk memperbaikinya tidak mahal. Rational Rose mendukung pemodelan bisnis, membantu kita memahami bisnis yang terjadi pada sistem, membantu analisis sistem dengan memungkinkan kita untuk merancang use case dan use case diagram untuk menunjukkan fungsionalitias dari sistem.

Rational Rose mempunyai keunggulan dasar dari sebuah rekayasa piranti lunak (*software engineering*), yaitu :

- Mendukung pengembangan sistem berbasis komponen dan
- Pengembangan secara interaktif dan konsistensi

Kedua konsep tersebut meskipun dikembangkan terpisah, tetapi digunakan secara bersama sehingga diperoleh manfaat pengembangan sistem yang optimal.
Rose menggunakan UML, COM (*Component Object Model*), OMT (*Object Modelling Technique*) dan *methode Booch 93* dalam memodelkan sistem secara visual. Penggunaan semantik, akan menjamin pengembangan sistem yang benar dan konsisten pada waktu pembangunan maupun perawatan sistemnya.

Rational Rose adalah software yang memiliki perangkat pemodelan secara visual untuk membangun suatu solusi rekayasa piranti lunak dan pemodelan bisnis. Rational Rose memakai UML sebagai bahasa pemodelannya ditambah beberapa fitur lain yang membuatnya menjadi software pemodelan visual yang terkemuka. Beberapa fitur diantaranya adalah Rational Rose memiliki *Rational Unified Process (RUP)* dan juga memiliki kemampuan membuat solusi *client/server* yang dapat diterapkan dan didistribusikan dalam lingkungan sebuah perusahaan.

a. Pemodelan Visual

Kompleksitas sistem yang meningkat karena berkembangnya lingkungan bisnis yang kompetitif, menjadi bagian hidup yang dihadapi sehari-hari oleh para developer. Model membantu untuk mengorganisasikan, memvisualisasikan, memahami dan membentuk sesuatu yang kompleks agar mudah dipahami.

Pemodelan visual adalah penggambaran proses-proses sistem pada dunia nyata dalam bentuk grafis. Model digunakan untuk memahami masalah, mengkomunikasikan dengan semua pihak yang berhubungan dengan sistem, seperti pelanggan, pemasok, tenaga ahli, sistem analis, programer, desainer dan lainnya; memodelkan sistem yang rumit, menyiapkan dokumentasi, merancang

program dan database. Dengan menggunakan model memungkinkan pemahaman *requirement* yang lebih baik, lebih lengkap, sehingga rancangan sistem akan lebih baik dan mudah untuk dirawat.
Sebuah sistem cenderung semakin kompleks. Untuk menggambarkan sistem secara efektif developer mulai dengan memperlihatkan gambaran besarnya sebelum ke detail. Model merupakan alat yang ideal untuk memotret abstraksi dari sebuah problema dengan memisahkannya dari hal-hal rinci yang dianggap tidak perlu.

Dalam praktek membangun sistem, seorang developer menggambarkan dalam cara pandang seperti layaknya seseorang membangun gedung. Ia meminta arsitek menggambar *blue-print* yang berisi gambar tata letak bangunan beserta fasilitasnya, seperti listrik, ruangan dan sebagainya; selanjutnya desainer akan menggambarkan sistem untuk memenuhi kebutuhan pemakainya.

Pemodelan visual menggunakan standar notasi UML. Ini memungkinkan transisi antara lingkungan (domain) bisnis ke lingkungan komputer. Dengan menggunakan UML semua anggota tim pengembang sistem dapat bekerja dalam bahasa yang sama, mengurangi miskomunikasi dan meningkatkan efisiensi. Pemodelan akan menggambarkan proses-proses bisnis dengan mendefinisikan kebutuhan sistem berdasarkan perspektif pengguna, hal ini mendukung aliran kerja dari desain ke proses pengembangan sistem.

Pemodelan visual juga menetapkan arsitektur dengan membuat arsitektur logika terhadap bahasa pemrograman. Metode yang fleksibel untuk menghubungkan arsitektur logik ke bahasa pemrograman. Pemodelan visual membantu penggunanan modul-modul yang dapat digunakan kembali (re-use) dengan membuat desain komponen. Selanjutnya komponen tersebut digunakan bersama dan *re-use* oleh pihak yang berbeda dalam pengembangan sistem.
Visual modeling adalah proses menggambarkan cetak biru suatu sistem secara grafis terdiri dari komponen-komponen, *interface* dan koneksi-koneksi yang ada dalam sistem tersebut, agar mudah dipahami dan dikomunikasikan.
Visual modeling dapat membantu untuk menampilkan elemen-elemen yang penting secara detail dari suatu masalah yang kompleks dan menyaring untuk kemudian membuang elemen-elemen yang tidak penting

Rational Rose menggunakan bahasa UML sebagai bahasa pemodelannya. Semua semantik dan notasi dalam UML dibuat untuk digunakan dalam visual modeling. Model yang dapat dikerjakan dengan UML ada dua yaitu model bisnis dan model untuk rekayasa software.

Agar model yang dibuat memenuhi kriteria sebagai model yang bagus, maka model yang dibuat harus dapat :

- Mengidentifikasikan persyaratan-persyaratan (requirement) dan dapat menyampaikan informasi dengan jelas
- Berfokus pada bagaimana komponen-komponen sistem berinteraksi
- Membantu untuk melihat hubungan antara komponen
- Meningkatkan komunikasi antara anggota tim pengembang sistem dengan menggunakan bahasa yang mudah dipahami dan dalam bahasa grafis

b. Pemodelan dengan Rational Rose

Rational Rose merupakan solusi perangkat lunak yang menggunakan model visual untuk meng-*create*, menganalisis, desain, view, modifikasi dan mengolah komponen-komponen. User dapat menggambarkan *overview* dari sistem dengan menggunakan *use case diagram*. Diagram ini untuk menggambarkan interaksi antara objek-objek dan hubungannya dengan sistem yang dibangun . Sedangkan *statechart diagram*, merupakan alat analisis terhadap *class* untuk menggambarkan perilaku dinamisnya. Statechart menggambarkan sejarah kehidupan sebuah *class*; sedangkan *event* akan menjelaskan transisi dari satu *state* ke yang lain dan aksi terhadap *state* yang ada.
Selanjutnya *activity diagram* untuk memodelkan operasional dari *class-class* yang ditampilkan dalam sebuah aliran kerja (*workflow*).

Rose menyediakan notasi-notasi untuk mendokumentasikan arsitektur dari sistem:

- Arsitektur logik (pada *class diagram*), yang berisi kelas-kelas dan relasi diantaranya.
- Arsitektur komponen (pada *component diagram*), menjelaskan bagaimana modul-modul sistem diorganisasikan didalamnya.
- *Deployment*, menggambarkan urutan proses di dalam sistem.

Rational Rose memiliki keunggulan, antara lain :

- Bahasa yang digunakan adalah bahasa pemodelan standar yaitu UML, sehingga meningkatkan komunikasi anatar anggota tim pengembang sistem
- Rational Rose mendukung *round trip engineering*, sehingga dapat men-*generate* dari model menjadi code (Java, C++, Visual Basic dan sebagainya) dan melakukan *reverse engineering* untuk menampilkan arsitektur software berdasarkan software code yang ada.
- Model dan code selalu sinkron selama *development cycle*
- Memudahkan dalam melakukan *maintenance system*, karena apabila suatu saat ditemukan *requirement* baru, maka dapat dengan cepat menggambarkan kembali software tersebut dengan UML
- Dapat bekerjasama dalam beberapa platform yang berbeda(Windows, Unix)
- Dapat mengkomunikasikan model dan spesifikasinya melalui *web browser*

- Mendukung rekayasa untuk sistem *client/server*.

c. Notasi

UML menyediakan notasi-notasi yang lengkap mulai dari analisis sampai dengan desain. Elemen-elemen pada notasi dipakai pada waktu analisis, terdiri dari *use-case, class, asosiasi, agregasi, inheritance* dan sebaginya.

Peran notasi tersebut adalah sebagai berikut :

- digunakan untuk komunikasi diantara berbagai komponen sistem
- menyediakan semantik yang menyimpan catatan strategik dari pilihan user
- memiliki form dan tool yang dapat dioperasikan secara bersama

d. Fitur-fitur pada Rose

Beberapa fitur pada Rational Rose yang digunakan selama analisis, desain dan konstruksi sistem adalah :

- use case analysis
- Object oriented modeling
- Support to UML, COM, OMT, Booch'93
- Semantic checking
- Mendukung pengembangan system secara iterative
- Round-trip engineering
- Mendukung pengembangan system oleh user secara parallel
- Terintegrasi dengan tool pemodelan data
- Mempunyai fasilitas untuk membuat dokumentasi
- Script yang terintegrasi dan ekstensif
- OLE linking
- OLE automation
- Ketersediaan sumberdaya dalam berbagai platform yang berbeda

e. Extended Rose dan Add In Manager

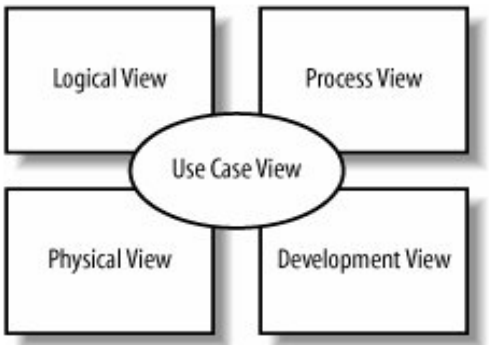
Tersedianya *tool add-in* untuk instalasi bahasa tertentu seperti Visual Basic, Java dan beberapa tool seperti Project Management. Sedangkan komponen *add-in* tersebut terdiri dari :

- Menu (.mnu file)
- help file (.hlp file)
- contents tab file (.cnt file)
- properties (.pty file)
- executables (.exe)
- script files (.ebs dan .ebx)
- OLE server (.dll)

Add-in manager berguna untuk melakukan control status dari add-in; apakah dalam kondisi aktif atau tidak aktif. Jika tidak aktif maka tampilannya tidak terlihat dan property serta menu-menunya tidak tersedia.

3.2. Views pada Rational Rose

Untuk melihat sistem dari berbagai aspek, RUP mendefinisikan atau membaginya menjadi 4 (empat) + 1 (satu) cara pandang atau *view.*



Tiap-tiap *vew* menjelaskan penegasan dalam aspek yang berbeda mengenai sistem yang kita modelkan. Berikut adalah penjelasan dari masing-masing cara pandang yang disediakan oleh Rational Rose :

3.2.1. Use Case View

Membantu untuk memahami dan menggunakan sistem yang kita modelkan. *View* ini melihat pada bagaimana *actor* dan *use-case* berinteraksi.

3.2.2. Logical View

Cara pandang yang mengarah pada persyaratan fungsional sistem, melihat pada kelas-kelas dan hubungan antar kelas-kelas.

3.2.3. Physical View

View yang menggambarkan bagaimana suatu sistem di *design* dan memperlihatkan pemetaan setiap proses kedalam *hardware..*

3.2.4. Component View

View yang mengandung informasi mengenai komponen-komponen *software*, komponen *executable*, dan *library* untuk sistem yang kita modelkan.

3.2.5. Deployment View

Menggambarkan bagaimana bagian-bagian dari sistem dikelompokkan menjadi modul-modul atau komponen-komponen.

3.3. Graphical User Interface (GUI) pada Rational Rose

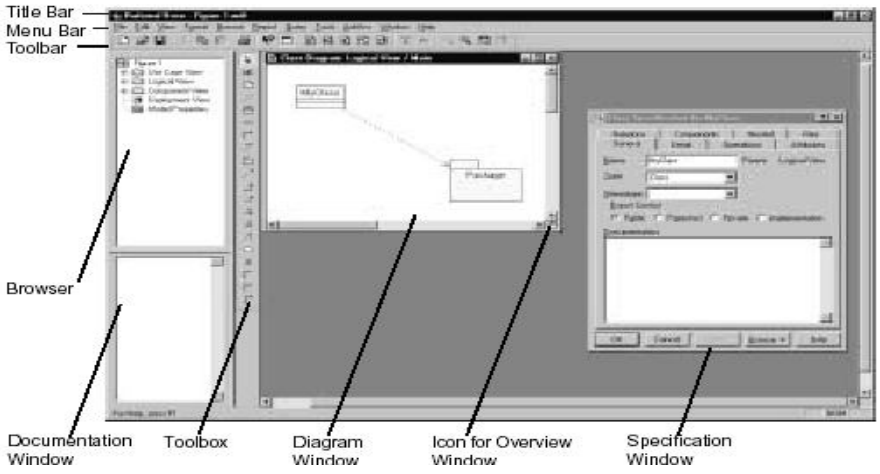
Pada saat awal mengaktifkan Rational Rose, akan ditampilkan framework dialog box. Dari box ini, dapat dipilih model dengan elemen-elemen yang telah didefinisikan sesuai dengan kebutuhansistem. Rose akan menampilkan sebagai berikut :

- Aplication window
- Browser
- Toolbox
- Documentation window
- Diagram window
- Overview window
- Specification window
- Log window

Menjalankan Rational Rose

- Jalankan Rational Rose dari **Start Menu- Programs-Rational Rose 2000 Enterprise Edition-Rational Rose 2000 Enterprise Edition**.
- Saat pertama kali menjalankan akan muncul kotak dialog **Create New Model**. Ikon-ikon yang terdapat pada jendela tersebut adalah framework yang sudah disediakan bagi pemakai Rose.

Figure 1 Application Window



Pada aplication window akan ditampilkan title bar, toolbar, work area yang berisi toolbox, browser, documentation, diagram dan spesification-nya

Title Bar :

- terletak di bagian paling atas layar
- menampilkan tipe dari diagram yang sedang aktif
- juga berisi control menu-box, minimize button, Restore dan Close

Untuk menampilkan control menu box

- Lakukan klik pada area control menu, maka box akan ditampilkan :
  Restore, Move, Size, Minimize, Maximize dan Close

Menu Bar :

Yaitu standar menu berupa simbol-simbol yang dapat dipilih untuk menggunakan diagram-diagram

Toolbar Standar :

Ditampilkan di bawah menu bar, di bagian atas aplikasi. Terdiri dari beberapa icon, yaitu :

Figure 2 Standard Toolbar

New Model : digunakan untuk membuat model sistem yang baru
Open Model : untuk membuka dan melakukan load model yang telah ada
Save model/log : Menyimpan model yang dibuat (dengan diberikan nama file)
Cut : menghapus simbol pada model (harus di select terlebih dahulu)
Copy : menduplikasi elemen model pada lokasi yang berbeda di diagram
Paste : men-copy icon yang dipilih / cut pada lokasi yang berbeda
Print Diagram : untuk mencetak diagram
Help : online help topic
View Doc : menampilkan jendela dokumentasi
Zoom In/Out : memperluas atau memperkecil tampilan diagram

Toolbox

Berisi tool untuk menggambar diagram. Toolbox pada diagram Rational Rose berubah sesuai dengan jenis diagram yang aktif. Jika sebuah diagram sedang aktif maka icon-icon pada toolbox akan ditampilkan dengan *title bar* berwarna biru.

Cara memilih icon pada toolbox :

- klik kanan, pilih customized
- klik dua kali (double click) pada toolbox
- klik **view** – **Toolbar** – **Configure**
- klik **Tool** - **Option**

Browser

Browser adalah alat navigasi yang secara hirarki membantu menampilkan nama-nama icon diagram-diagram use case, class, interaction, statechart, activity, deployment dan lain-lain.
Tanda (+) menandakan elemen tersebut mengandung informasi tambahan didalamnya. Dengan menekan tanda (+), informasi tersebut diperluas. Sebaliknya tanda (-) menandakan informasi tersebut terbuka secara penuh. Jika browser tidak muncul, pilih **Browser** dari menu **View**
Jika sebuah class atau interface telah dipilih pada komponen, maka browser akan menampilkan nama komponen tersebut dan ekstensinya.
Ekstensi dari nama adalah, suatu daftar nama yang dipisahkan dengan tanda koma dalam kurung di bagian kanan dari nama class tersebut. Daftar ekstensi, berisi semua komponen yang dipilih

Documentation window
Digunakan untuk menjelaskan elemen rincian model dan relasinya. Deskripsi dapat berisi berbagai informasi tentang *Role, Key, Constraint, jenis dan property* elemen diagram yang esensial.
Dokumentasi dapat diisi melalui jendela *documentation* atau melalui kolom pada isian mengenai *specification*
Untuk menampilkan documentation : Click **View** – **documentation**
Hanya satu dokumentasi yang dapat ditampilkan pada suatu saat yang sama

Log Window
Menampilkan catatan-catatan (log) yang melaporkan kegiatan, hasil, error, selama pembuatan model. Penjelasan tersebut disimpan dengan catatan waktu da kejadiannya.
Untuk menampilkan log window, lakukan :

View – **Log**
File – save log As (untuk menyimpan catatan)

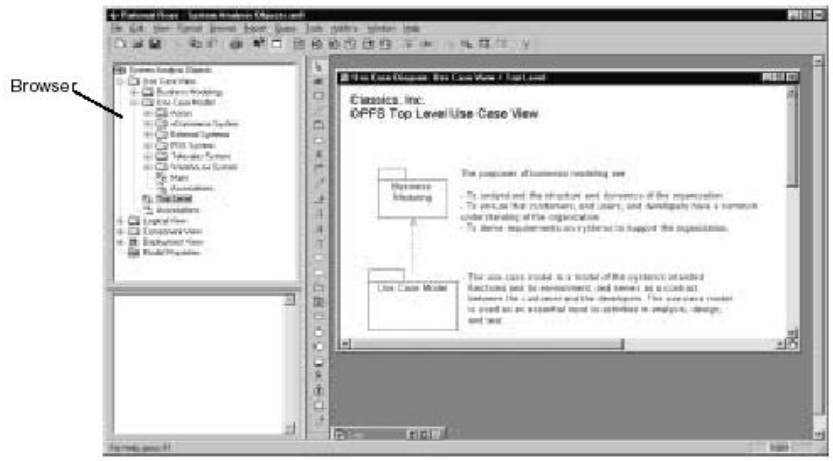
Diagram Window
Memungkinkan kita membuat dan melakukan modifikasi tampilan diagram dari sebuah model.
Kita dapat menyajikan sejumlah diagram (multiple diagram), setiap elemen model dapat nol, satu, atau banyak diagram
Sedangkan diagram yang ada pada Rose dikelompokkan menjadi :

Logical Package
Component
Class
Three tierd : deployment, logical, component

2. Browser

Browser merupakan cara yang mudah digunakan untuk menampilkan menu, toolbar dalam membuat visualisasi dan navigasi pembuatan model

Figure 3 Application Window



Browser berisi :

- tampilan secara hirarki dari model
- kemampuan memakai diagram dengan drag and drop
- melakukan pemutakhiran secara otomatis item-item dari model untuk menampilkan perubahan browser

a. View

Pada tampilan awal dari Rational Rose, browser akan diperlihatkan di bagian sebelah kiri dari diagram toolbox

Untuk menampilkannya :

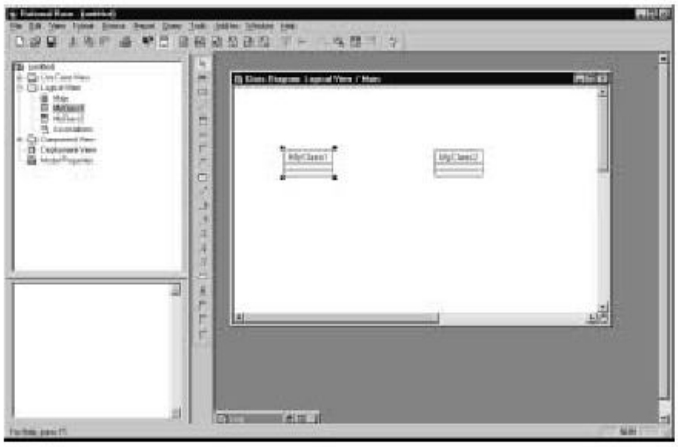
View – Browser, lalu pilih (*check mark*) untuk menampilkan atau *hide*.

Untuk mengubah ukuran dapat menggunakan *docked* atau *float*. Sedangkan *default* tampilannya adalah *docked*. Cara mengubah tampilan adalah dengan click di sebarang tempat pada browser, lalu *drag browser* pada tempat lainnya.

b. Navigasi

Browser menyediakan tampilan secara hirarki dari suatu model.

Figure 4 Navigating a Model



Untuk melakukan sinkronisasi diagram :

- tampilkan jendela diagram, lakukan double click nama atau icon pada browser
- menampilkan specification, lakukan double-click item pada browser atau pada jendela diaggram
- untuk menyorot item diagram, click item tersebut pada browser atau pada diagram
- *Expand* atau *collapse tree*
- tanda (+) menyatakan collapse, artinya berisi diagram-diagram yang lain
- tanda (-) menyatakan icon tersebut telah di expand

Figure 5 Browser—Collapsed and Expanded Tree

Membuat dan mengedit elemen :

- dapat dilakukan pada browser dengan cara (+) atau (-) item di dalamnya dan secara otomatis akan meng-update diagram
- drag and drop elemen browser ke satu diagram tertentu
- jika class diagram mempunyai *parent* dan *show visibility On* maka akan terlihat lokasi dari Class tersebut.

Memberi nama elemen browser

- *create* atau *select* elemen
- ketikkan nama elemen tersebut

Memilih multiple elemen
Berguna untuk membuat diagram pada waktu melakukan control system

- pilih item *browser*
- tekan *Ctrl Key*
- pilih item lain pada *browser*

atau

- pilih item *browser*
- tekan *shift key*
- pilih item lain pada *browser*

Mengurutkan paket
(1) *Create new package*, lalu berikan nama *Temp*
(2) Pada *browser*, drag and drop semua paket yang akan di sort
(3) Pada *browser*, ambilah satu persatu dari temp lalu tempatkan pada lokasinya
(4) *Delete* temp

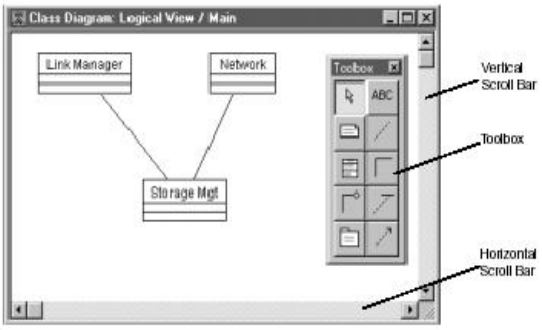
c. Using Drag and Drop
Dipakai untuk memindahkan item-item dari browser ke diagram dan specificationnya
Penggunaan drag and drop adalah untuk melakukan hal-hal sbb :

- memilih *class* dan *component interface*
- memindah *class operation* dan atribut diantara *class*
- memindahkan *class, sequence, collaboration* diagram antar paket
- memindahkan *component* antara paket
- memindahkan *nested class* pada satu specification ke yang lain
- untuk menempatkan komponen dan paket dalam satu *package*
- menempatkan *class, interface, component* ke class diagram
- menempatkan *object*, isi *class* pada diagram interaction
- memindahkan komponen atau *parent* diantara *component package*
- memindahkan *class, nested class, use case, interface, asociation* diantara package
- menempatkan elemen *activity* diagram pada diagram activity

3.4. View pada diagram-diagram Rational Rose

Diagram menampilkan informasi yang ada pada sebuah model. Rational Rose akan menjamin konsistensi diantara diagram-diagram dan spesifikasinya
Untuk mengubah model dapat, dapat dilakukan melalui perubahan pada properti atau relasi pada specification atau pada icon diagram.
Jika dilakukan perubahan, maka diagram-diagram yang berhubungan akan secara otomatis di-update

Figure 6 Diagram Window



a. Jendela Diagram :

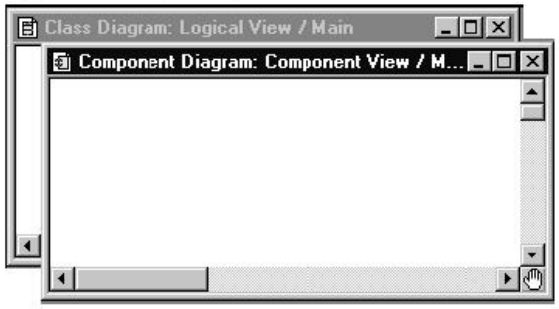
Terdapat delapan macam diagram untuk menggambarkan sistem yaitu

- *Class* Diagram
- *Use case* diagram
- *Collaboration* diagram
- *Sequence* diagram
- *Component* diagram
- *Statechart* diagram
- *Deployment* diagram
- *Activity* diagram

View diagram :

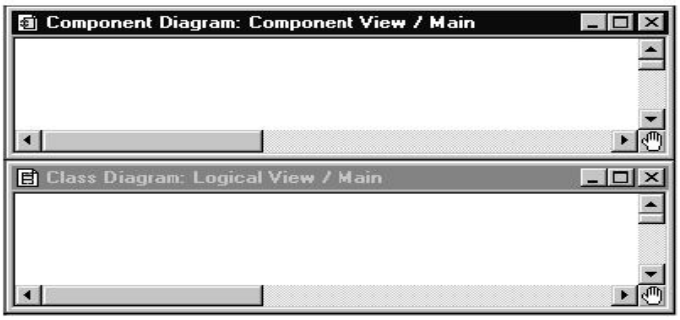
Diagram dapat di buka (*open)*, disembunyikan *(hide*), diperluas atau di tutup (*closed)*

Figure 7 Multiple Diagrams—Cascade Windows



Juga dapat membuka beberapa diagram sekaligus (*multiple diagram*), dengan memilih tampilan *cascade* atau *tile.*

Figure 8 Multiple Diagrams—Tiled Windows



b. Membuat diagram

- **browse** > **xxx Diagram** (xxx : nama diagram)
- click <**new**>
- click **OK**
- Berikan nama diagram
- **OK**

Linking :

- *create notes* pada diagram
- display *browser*
- tempatkan diagram pada browser
- *drag icon* diagram asal ke diagram tujuan
- ubah teks
- *double click*

Display diagram, Rename dan Delete Diagram

- **Browse > xxx Diagram**
- Select Diagram
- **OK**

c. Membuat elemen diagram

Create elemen :

- click *creation too*l
- click lokasi diagram

Create elemen browser

- Click *package*
- Click *New*
-

Memberi nama elemen

Dapat diberi nama yang bermakna dan dapt digunakan kembali

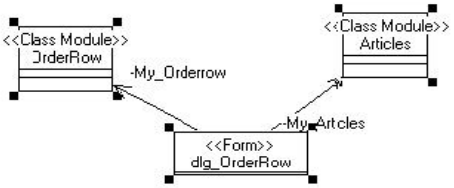
- create *new elemen*
- ketik nama elemen

Re-assigning model

- select *icon*
- click *edit* > *reassign*

d. Manipulasi ikon

Figure 9 Selected Elements in a Diagram



Select icon

- click kiri ikon
- tekan dan tahan *Ctrl/Shift* bersama click ikon yang dipilih

Deselect icon

- click pada sebarang tempat di area diagram

Resize

- click ikon
- click select *handle, drag* kiri ke ukuran yang diingkan

Memindahkan ikon, cut, copy and paste

- pilih ikon
- drag kiri
- *left button*

e. Menghapus elemen diagram

Shallow del

Yaitu menghapus elemen yang akan dihilangkan dari diagram

- click *edit* > del
- tekan *Del*

Deep del

Yaitu menghilangkan seluruh model dan elemennya

- click *edit* > *del* from model
- *ctrl* + D
- click kanan, lalu tekan *Delete*

f. Korelasi diantara diagram

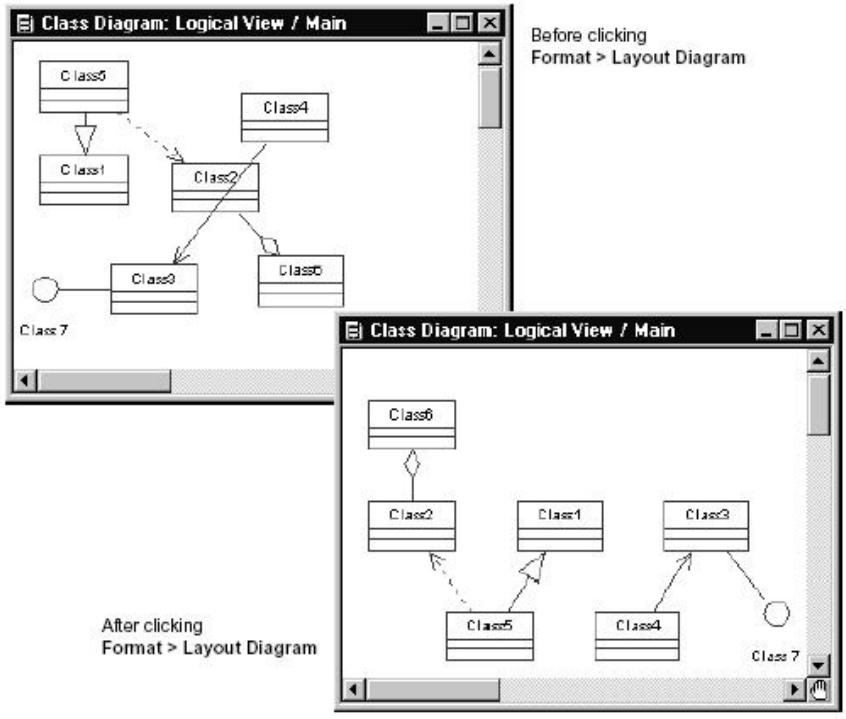
Diantara diagram dapat saling korelasi, berupa relasi, link, dependency, transisi atau koneksi

- click relasi pada *toolbox*
- pilih *client icon*
- tekan dan tahan *mouse* sebelah kiri
- drag ke diagram penerima
- lepaskan *mouse*

g. Laying diagram

Jika gambar pada diagram, yang terdiri dari beberapa shape dan relasi menjadi sulit dibaca, maka dapat diatur kembali dengan mengurangi sehingga sedikit mungkin terjadi *crosslink*, maka hal ini disebut laying diagram.

Figure 10 Example Layout of a Class Diagram



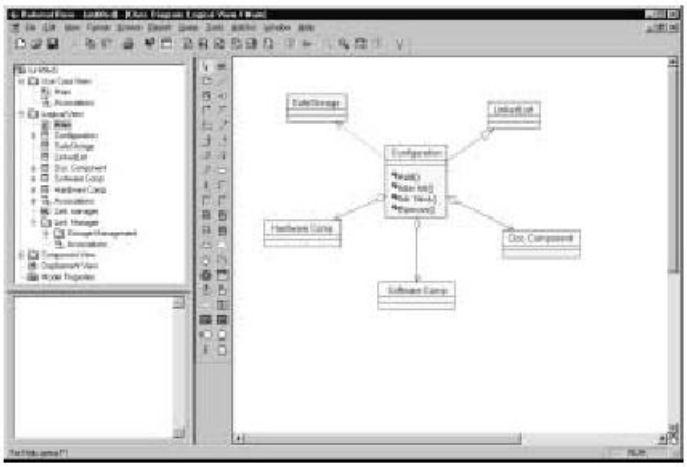
- click *tool > option*
- pada general tab atur pilihan pada *lay-out option*
- click *OK*
- click *format > lay-out diagram*

3. Class Diagram

Class diagram menggambarkan pemodelan objek. Berisi *class, object* serta dapat menggambarkan keduanya dalam satu metadata.

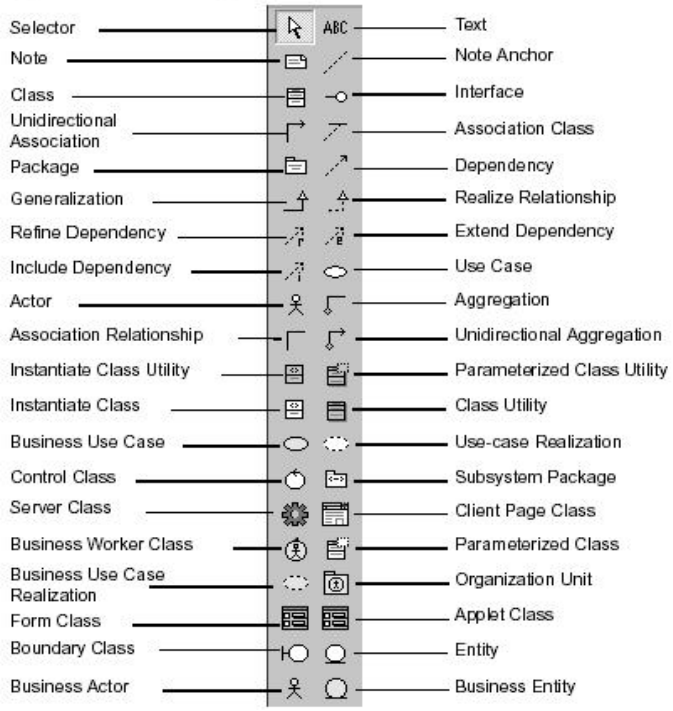
Class diagram berisi icon-icon yang menyatakan class, interface, relasi. Kita dapat membuat lebih dari satu class diagram yang dinyatakan dalam satu package, sementara class diagram tersebut dapat berisi setiap paket dari model.

Figure 16 Class Diagram Example



Class Diagram Toolbox :

Figure 17 Class Diagram Toolbox



Untuk membuat dan menampilkan class diagram, dapat menggunakan tiga cara, yaitu :

Click **browse** > **class diagram**, atau

Pada *Toolbar*, *click icon class* diagram, atau
Pada *browser*, lakukan *duble click icon class* diagram

Setiap *class* ditempatkan (assigned) pada satu paket (logical package). Jika sebuah *class* dibuat menggunakan *toolbox* diagram, maka otomatis class tersebut ditempatkan pada package tadi.
Sebagai contoh, sebuah class diagram dengan nama Main otomatis akan berisi package *LinkManager.* Maka semua class yang digambarkan pada Main akan ditempatkan pada package *LinkMaager* itu.

Untuk mengatur penempatan kembali class dari satu logical package, dapat dilakukan dengan :

- select *icon* yang menyatakan *class diagram* yang berisi package logical ke dalam class yang ditunjuk
- click **Edit** > **Relocate;** maka Rose akan meng-update semua class diagram yang menyatakan perintah tersebut dijalankan.

Dapat juga melakukan penambahan, menyembunyikan (*hide*) dan memilih (*filtering*) relasi class, untuk itu dapat dilakukan pada menu Query :

- *Add Class, Add Use Case, Expand Selected Elements* untuk menambahkan class, use case dan elemen ke dalam diagram
- *Hide Slected Elements* untuk menyembunyikan elemen diagram
- *Filter relationship* untuk kontrol terhadap relasi antar diagram

4. Use Case Diagram

Use Case diagram menggambarkan tampilan level atas sebuah sistem yang dibangun yang dilihat dari perspektif *user* atau *actor*. Use case diagram juga menyatakan perilaku dari sistem, karena berisi semua use-case yang ada pada sistem tersebut.

Komponen dari use case diagram :

- *Actor,* adalah semua yang ada diluar system
- *Use Case*, adalah batas-batas system yang mengidentifikasi apa saja yang dikerjakan oleh system
- *Interaksi,* atau relasi antara actor dengan use case

Use case diagram biasanya dipakai selama tahap analisis untuk mencatat keperluan (*requirement*) sistem dan bagaimana sistem seharusnya bekerja
Sedangkan pada waktu desain, use case diagram berguna untuk membuat rincian perilaku dari cara implementasi sistemnya

Untuk membuat dan menampilkan use case diagram, digunakan tiga cara, yaitu :

Click **browse** > **use case diagram**, atau

Pada *Toolbar*, *double click icon use case* diagram, atau

Pada *browser,* lakukan *double click icon use case* diagram

a. Actor

Menyatakan pengguna (user); mereka membantu mendefinisiskan sistem, memberikan gambaran yang jelas apa yang harus dikerjakan sistem. Perlu diperhatikan, bahwa Actor berinteraksi dengan sistem tetapi tidak ikut mengatur cara operasi dari use case.

Actor adalah seseorang yang :

- berinteraksi atau menggunakan sistem
- memberikan masukan(input) ke atau menerima informasi dari sistem
- diluar sistem dan tidak ikut mengkontrol use case

Untuk menemukan aktor dengan beberapa petunjuk :

- siapakah yang menggunakan sistem
- siapakah yang bertanggungjawab terhadap sistem
- perangkat lain di luar yang menggunakan sistem
- sistem informasi lain yang berinteraksi dengan sistem

b. Use Case

Adalah urutan kejadian (transaksi) yang dilakukan oleh system sebagai reaksi (respon) terhadap masukan yang diberikan oleh *Actor.*

Use case menjelaskan seluruh kejadian(*event)* yang muncul dari pasangan actor dengan use case.

Use case menggambarkan cara khusus, bagaimana sistem bekerja dilihat dari perspektif aktor, yang meliputi :

- pola dari perilaku sistem
- urutan transaksi-transaksi yang dilakukan oleh aktor dan sistem

Use case juga berguna untuk mencatat keperluan(*requirement*) sistem, untuk komunikasi dengan pengguna(*user)* dan berguna untuk melakukan test sistem

c. Flow of Event

Adalah urutan dari transaksi yang dijalankan oleh sistem. Biasanya diuraikan secara rinci, untuk menggambarkan apa yang dikerjakan oleh sistem. Biasanya *flow of event* dibuat pada satu file terpisah dengan memakai text editor dan dihubungkan dengan use case sebagai satu file pada elemen model.

Flow of event, menjelaskan :

- kapan dan bagaimana use case dimulai dan berakhir
- menggambarkan interaksi antara use case dan aktor
- menampilkan data yang diperlukan oleh use case
- urutan normal dari event di dalam use case
- pilihan aliran atau pengecualian

Untuk menggambarkan flow of event ini dapat digunakan ***Activity Diagram***.

d. Relationship

Relationship menggambarkan interaksi antara *actor* dengan *use case*. Asosiasi, dependensi dan generalisasi dapat digambarkan dari *actor* ke *use case*. Sedangkan generalisasi dapat digambarkan antara aktor.
Setiap asosiasi dinyatakan dalam format teks pada tampilan relasinya

e. Asosiasi

Asosiasi memungkinkan hubungan yang mengalir antara *use case* dengan aktor. Asosiasi adalah hubungan yang sangat umum; jika dua buah objek tidak saling bergantung, maka hubungannya disebut asosiasi. Pemberian nama asosiasi harus mencerminkan arti hubungannya.
Terdapat dua jenis asosiasi, yaitu :

- *Uni-directional association*, yaitu asosiasi antara use case yang dinyatakan dengan tanda panah diantara use case. Use case yang terletak di ujung kepala panah, adalah yang menerima hasil komunikasi.
- *Bi-directional association*, menggambarkan hubungan timbal balik (saling memberikan komunikasi) anatar dua buah use-case. Untuk membuat hubungan bi-directional ini, lakukan double click pada asosiasi lalu pilih **Assosiation Specification**. Click salah satu **Role A** atau **Role B** pada **Detail tab**, lalu pilih **Navigable** check box selanjutnya tekan **Apply**.

f. Dependency

Dependensi adalah hubungan antara dua elemen model yang mana satu elemen model mempengaruhi elemen yang lain. Biasanya dipakai untuk hubungan dua model yang mempunyai level yang sama, seperti pada class diagram dimana permintaan seorang client tergantung kepada pemasok.
Untuk menggambarkan dependensi ini, digunakan *state machine diagram* dan *object diagram*.

g. Generalisasi

Hubungan yang terjadi karena terdapat elemen use case atau elemen class yang bersifat umum (general) yang harus dipisahkan dari bagian yang spesifik. Digambarkan dengan tanda panah dari spesifik elemen ke elemen yang lebih

umum (general). Untuk membuat rincian stereotipnya, dapat dibuat melalui **generalization specification**.

Reference :

1. support@rational.com, http://www.rational.com
2. USING ROSE Rational Rose®, Version : 2001A.04.00, Part Number : 800-024462-000
3. Wendy Boggs and Michael Boggs, UML with Rational Rose 2002, Sybex(2002)

LATIHAN:

1. Sebutkan tampilan-tampilan yang muncul pada *Graphical User Interface* (GUI) *Rational Rose.*
2. Sebutkan diagram-diagram yang ditampilkan pada jendela diagram *Rational Rose.*
3. Jelaskan tentang *view* yang ada pada *Rational Rose*.