MODUL REKAYASA PERANGKAT LUNAK (CCR 210)

MODUL 14
ANALISIS DAN DESIGN OBJECT ORIENTED

DISUSUN OLEH
HANI DEWI ARIESSANTI, M.KOM

UNIVERSITAS ESA UNGGUL
2019

ANALISIS DAN DESAIN BERORIENTASI OBJECT

A. Pendahuluan

Analisis adalah Mempelajari domain permasalahan bisnis untuk merekomendasikan perbaikan dan menentukan kebutuhan system untuk menyelesaikan permasalahan.

Desain adalah Menentukan solusi teknis/computer-based dimana solusi ini berdasarkan kebutuhan system yang telah diidentifikasi pada proses analisis.

OOA (Object Oriented Analisis) adalah metode analisis sistem yang biasanya dimulai dengan adanya dokumen permintaan (*requirement*) yang diperoleh dari semua pihak yang berkepentingan. (Mis: klien, developer, pakar, dll). Dokumen permintaan memiliki 2 fungsi:

1. Memformulasikan kebutuhan klien
2. Membuat suatu daftar tugas

Ada beberapa pendekatan yang biasa digunakan dalam menganalisa suatu sistem, yaitu:

1. *Functional Decomposition*
   - Functional Decomposition = function + sub-functions + functions interfaces
   - *Functional decomposition* hanya menghasilkan suatu spesifikasi fungsional yang memetakan subyek secara tidak langsung.
   - *Functional decomposition* sulit dilakukan karena sifat sistem yang mudah berubah. Oleh sebab itu sistem analist harus dapat menentukan fungsi-fungsi yang memiliki potensi untuk berubah.
2. *Data Flow Approach*
   - Data Flow Approach = data (& control) flows + data (& control) transformation + data (& control) stores + terminators + process specs (mini-specs) + data dictionary
   - Strategi: Mengikuti alir data yang terjadi dalam sistem.
   - Yang menjadi masalah adalah, manusia tidak terbiasa berpikir dengan cara tersebut dalam memecahkan permasalahan (Metode pengorganisasian).

3. *Object Oriented*
   - Object Oriented = classes and objects
   - + inheritance
   - + communication with messages

OOD (Object Oriented Design) adalah metode untuk mengarahkan pada arsitektur Software yang didasarkan pada manipulasi objek-objek system atau subsitem. Arsitektur software menjelaskan susunan sistem yng terdiri dari komponen software, atribut dari komponen dan yang ada hubungan antar komponennya. Komponen dapat berupa modul, database, middleware, atau class. Atribut adalah ciri dan fungsi modul. Hubungan antar komponen adalah cara antar komponen tersebut berkomunikasi, sperti modul satu memanggil modul lain.

OOP (Object Oriented Programming) adalah

B. Kompetensi Dasar

Mahasiswa mampu menganalisa dan mendesain sistem yang berbasiskan pada prinsip-prinsip obyek.

C. Kemampuan Akhir Yang Diharapkan

1) Mahasiswa mampu menjelaskan latar belakang perlunya metode berorientasi objek (OOA)
2) Mahasiswa mampu menjelaskan konsep, model dan penggunaan metode berorientasi objek dalam pengembangan system.
3) Mahasiswa mampu menjelaskan objek dan class
4) Mahasiswa dapat menjelaskan siklus sistem (OODLC) Object Development life Cycle
5) Mahasiswa mampu menjelaskan requirement Model
6) Mahasiswa mampu mengerti class, hierarchies dan agregasi
7) Mahasiswa mampu membuat diagram-diagram UML
8) Mahasiswa mampu mengimplementasiksan konsep untuk mendesain sistem yang berorientasi objek

D. Kegiatan Belajar

Pengertian Analisa Dan Desain Berorientasi Objek
Object Oriented Analysis And Design (OOAD)

Berorientasi objek berarti bahwa kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya.
OOAD (Object Oriented Analysis And Design) mencakup analisis dan desain sebuah sistem dengan pendekatan objek. Sehingga, dapat diartikan pula *OOAD (Object Oriented Analysis And Design)* adalah paradigma baru tentang perangkat lunak berdasarkan abstraksi yang terdapat dalam dunia nyata. Dalam konteks pengembangan menunjuk pada bagian awal dari siklus hidup pengembangan sistem, yaitu survei, analis, desain, implementasi dan pemeliharaan sistem. Hal yang lebih penting dalam pengembangan berorientasi objek adalah konsep mengidentifikasi dan mengorganisasi domain aplikasi dari pada penggunaan bahasa pemrograman, berorientasi objek atau tidak.
Untuk penggambaran tersebut diperlukan pemahaman dasar dari beberapa konsep dasar dalam *OOAD (Object Oriented Analysis And Design)*, yaitu:

a. Class And Object

Class adalah definisi umum (pola, template, atau cetak biru) dari himpunan objek yang sejenis. Kelas menetapkan spesifikasi perilaku (behaviour) dan atribut-atribut dari objek tersebut. Class adalah abstraksi dari entitas dalam dunia nyata. Sedangkan objek adalah contoh *("instances")* dari sebuah kelas.

Misalnya, atribut dari kelas binatang adalah berkaki empat dan mempunyai ekor. Perilakunya adalah makan dan tidur. Sedangkan contoh *(instance)* untuk kelas binatang ini adalah kucing, gajah, dan kuda.

Objek adalah benda secara fisik dan konseptual yang ada di sekitar kita. Beberapa contoh objek, misalnya hardware, software, dokumen, manusia, konsep, dan lainnya. Untuk kepentingan pemodelan, misalnya seorang

eksekutif akan melihat karyawan, gedung, divisi, dokumen, keuntungan perusahaan sebagai sebuah objek. Sedangkan seorang teknisi mobil, akan melihat ban, pintu, mesin, kecepatan tertentu dan banyaknya 16 bahan baker sebagai sebuah objek. Contoh lainnya adalah seorang software engineer akan memandang tumpukan, antrian intruksi, window, check box sebagai sebuah objek.

Sebuah objek mempunyai keadaan sesaat yang disebut *state.*

State dari sebuah objek adalah kondisi dari objek itu atau himpunan keadaan yang menggambarkan objek tersebut. Sebagai contoh, state dari rekening tabungan, dapat memuat saldo yang berjalan, state dari sebuah jam adalah catatan saat itu; sedangkan state dari sebuah bohlam lampu adalah suatu keadaan “nyala” atau “mati”. State dinyatakan dengan nilai dari atribut objeknya.

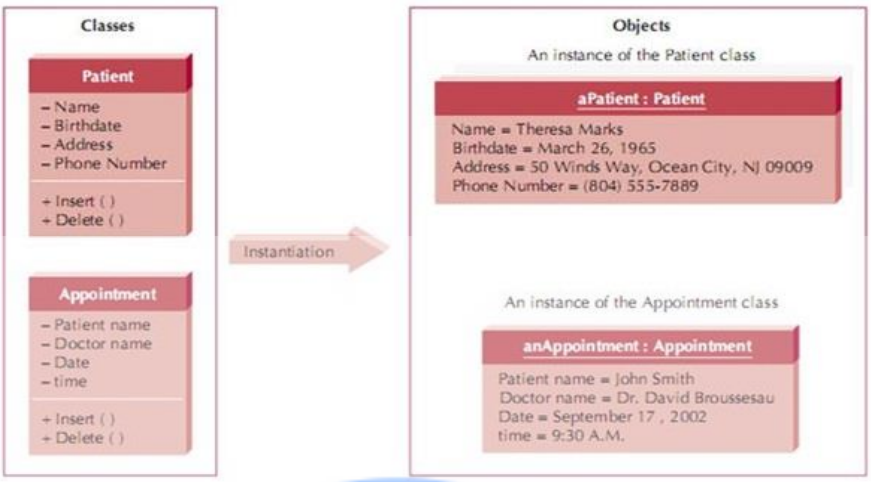
Atribut adalah nilai internal suatu objek yang mencerminkan antara lain karakteristik objek, kondisi sesaat, koneksi dengan objek lain dan identitas. Perubahan state dicerminkan oleh perilaku (behaviour) objek tersebut.

Behaviour atau perilaku sebuah objek mendefinisikan bagaimana sebuah objek bertindak(beraksi) dan memberi reaksi. Behaviour ditentukan oleh himpunan semua atau beberapa operasi yang dapat dilakukan oleh objek itu sendiri. Behaviour dari sebuah objek dicerminkan oleh interface, service dan method dari objek tersebut.

Interface adalah pintu untuk mengakses service dari objek.

Service adalah fungsi yang dapat dikerjakan oleh sebuah objek.

Abstraksi prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi suatu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan masalah



Gambar: Class And Object

b. *Method Dan Messages*

Method adalah mekanisme internal objek yang mencerminkan perilaku(behaviour) objek tersebut.

Message adalah permintaan agar objek menerima (receive) untuk membawa metode yang ditunjukkan oleh perilaku dan mengembalikan result dari aksi tersebut kepada objek pengirim (sender). Contohnya, satu objek orang mengirim kepada objek bola lampu sebuah pesan message untuk menyala melalui saklar. Objek bola lampu memiliki perilaku yang akan mengubah keadaannya(state) dari padam menjadi menyala. Objek lampu menyalakan dirinya dan menunjukkan kepada objek orang tersebut bahwa state barunya adalah menyala.

c. *Black Boxes Dan Encapsulation*

Sebuah objek adalah kotak hitam (black-boxes). Konsep ini menjadi dasar untuk implementasi objek. Dalam operasi OO, hanya para developer (programmer, desainer, analis) yang dapat memahami detail dari proses-proses yang ada didalam kotak hitam tersebut, sedangkan para pemakai (user) tidak perlu mengetahui apa yang dilakukan, tetapi yang penting mereka dapat menggunakan objek untuk memproses kebutuhan mereka. *Encapsulation*, proses menyembunyikan detail implementasi sebuah objek. Satu-satunya jalan untuk mengakses data objek tersebut adalah melalui interface. Interface melindungi internal state sebuah objek dari “campur tangan” pihak luar. Oleh karena itu objek digambarkan sebagai sebuah kotak

hitam yang menerima dan mengirim pesan-pesan (messages). Dalam OOP kotak hitam tersebut berisi kode (instruksi yang dipahami computer) dan data (informasi dimana instruksi tersebut beroperasi dengannya). Dalam OOP kode dan data disatukan dalam sebuah “benda” yang tersembunyi isinya yaitu objek. Pengguna objek tidak perlu mengetahui isi dalam kotak tersebut; untuk berkomunikasi dengan objek, diperlukan pesan(message).

d. ***Inheritance***

Inheritance atau pewarisan digunakan untuk mengidentifikasi object/class yang memiliki tingkat yang lebih tinggi dan lebih general. Atributes dan methods yang sama dikelompokkan menjadi sebuah superclass. Biasanya superclass diletakkan diatas dan turunannya diletakkan dibawah. Hubungan inheritance yang terjadi adalah A KIND OF. Subclass akan mewarisi attributes dan methods dari class yang berada diatasnya. Artinya subclass berisi attributes dan methods dari superclass yang berada diatasnya. Class yang berada didalam rantai hierarchy pasti akan ada yang memiliki instance/contoh kejadian. Class yang memiliki contoh kejadian disebut *concrete class*. Ada juga class yang menjadi superclass yang menjadi template bagi sebagian dari class yang berada dibawahnya. Class tersebut merupakan abstract class.

Antar muka biasanya digunakan agar kelas yang lain tidak mengakses langsung ke suatu kelas.

Reusability pemanfaatan kembali objek yang sudah didefenisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

Generalisasi/spesialisasi adalah atribut dan perilaku yang sama untuk beberapa kelas dikelompokkan ke dalam kelas tersendiri yang disebut supertype.

Supertype adalah entitas yang berisi atribut dan perilaku yang sama untuk satu atau lebih subtipe kelas. Juga disebut parent class.

Subtype adalah kelas yang mewarisi (inherit) atribut dan perilaku dari kelas supertype serta mungkin juga berisi atribut dan perilaku lain yang khusus. Juga disebut kelas anak (child). Jika berada pada tingkat yang paling rendah dalam hirarki pewarisan, akan disebut kelas konkret (*concrete)*

Komunikasi Antar Objek dilakukan lewat pesan (message) yang dikirim dari satu objek ke objek lainnya.

e. ***Polymorphism***

yaitu kemampuan suatu object untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama, sehingga menghemat baris program. Dibawah ini adalah contoh polymorphism :

1. *Polimorphism* pada C++ dapat dikenakan pada fungsi atau operator dan di kenal dengan *overloading*.
2. Berikut fungsi dengan nama gambar :
   - gambar(x,y); Menggambar titik
   - gambar(x1,y1,x2,y2); Menggambar garis
   - gambar(x,y,5); Menggambar lingkaran

Polymorphism dapat terlaksanya karena adanya *dynamic binding*. Dynamic, atau, late binding adalah teknik/cara penundaan penentuan jenis object sampai pada saat run-time. Artinya method mana yang akan dijalankan ditunda sampai system/program berjalan. Kontrasnya adalah static binding dimana setiap methods sudah ditentukan pada saat kompilasi. Package sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompok kelas-kelas yang bernama sama disimpan dalam package yang berbeda.

f. ***Asosisasi dan Agregasi***

Asosiasi adalah hubungan yang mempunyai makna antara sejumlah objek. Asosiasi digambarkan dengan sebuah garis penghubung di antara objeknya. Contoh :

Asosiasi antara objek mobil dengan seseorang. Mobil dapat dimiliki oleh satu atau beberapa orang, sedangkan seseorang dapat mempunyai nol, satu atau banyak mobil 18 Asosiasi antara karyawan dengan unit-kerja. Seorang karyawan bekerja di satu unit-kerja. Sedangkan sebuah unit-kerja dapat memiliki beberapa orang karyawan

Agregasi adalah bentuk khusus sebuah asosiasi yang menggambarkan seluruh bagian pada satu objek merupakan bagian dari objek yang lain (◊). Contoh :

Kopling dan piston adalah bagian dari mesin. Sedangkan mesin, roda, body adalah merupakan bagian dari sebuah mobil.

Tanggal, bulan dan tahun adalah bagian dari tanggal-lahir. Sedangkan tanggal-lahir, nama, alamat, jenis kelamin adalah bagian dari identitas Seseorang

Metedologi pengembangan sistem berbasis object

Metodologi adalah cara systematis untuk mengerjakan analisys and design. Dengan metodologi, pihak yang membangun system software dapat merencanakan dan mengulangi pekerjaan dilain waktu. Metodologi juga menghilangkan perbedaan notasi untuk suatu hal yang sama karena setiap orang akan berbicara dalam bahasa yang sama. Metodologi yang paling banyak dalam OOAD, yaitu : *Object Modeling Technique (OMT)* dari *Rumbaugh, Object Oriented Booch, Responsibility-Driven Design/ Class Responsibility Calloboration (RDD/CRC)* dari *Wirf-Broock, Metodologi Coad / Yourdan* dan *Jacobson Object Oriented Software Enginering (OOSE).*

1. *Object Modeling Technique (OMT)* Dikembangkan oleh *James Rumbaugh* sebagai metode untuk mengembangkan sistem berorientasi objek dan untuk mendukung pemograman berorientasi objek
2. *Object Oriented Booch* Dikembangkan oleh *Grady Booch* terdiri dari diagram kelas, objek, transisi status, interaksi, modul dan proses.
3. *Class Resposibility Calloboration (CRC)* Merupakan bagian dari *Object-Oriented Programming,System,Languages And Application(OOPSLA).* Dibuat untuk menjadi kelas yang akan dianalisis.
4. Metodologi *Coad / Yourdan* Menyediakan sebuah diagram kelas, pembuatannya dengan langkah-langkah berkut :
    - Mendefenisikan kelas dan objek
    - Mengidentifikasi struktur kelas dan objek.

- Mendefenisikan subjek nama kelas.
- Mendefenisikan atribut.
- Mendefenisikan operasi / layanan (service).

5. *Object Oriented Software Enginering (OOSE)* Dikembangkan oleh *Ivar Jacobson* adalah metode disain berorientasi objek yang melibatkan use case.

Teknik pemodelan yang ada pada OOAD

1. Model Objek :

- Model objek Menggambarkan struktur statis dari suatu objek dalam sistem dan relasinya
- Model objek berisi diagram objek. Diagram objek adalah graph dimana nodenya adalah kelas yang mempunyai relasi antar kelas.

2. Model Dinamik

- Model dinamik menggambarkan aspek dari sistem yang berubah setiap saat.
- Model dinamik dipergunakan untuk menyatakan aspek kontrol dari sistem.
- Model dinamik berisi state diagram. State diagram adalah graph dimana nodenya adalah state dan arc adalah tarnsisi antara state yang disebabkan oleh event.
- Model Fungsional
- Model fungsional menggambrakan transformasi nilai data di dalam sistem.
- Model fungsional berisi data flow diagram. DFD adalah suatu graph dimana nodenya menyatakan proses dan arcnya adalah aliran data.

ALASAN MENGGUNAKAN OOAD

1. Memudahkan pemanfaatan ulang code dan arsitektur
2. Lebih mencerminkan dunia nyata (lebih tepat dalam menggambarkan entitas perusahaan, dekomposisi berdasarkan pembagian yang natural, lebih mudah untuk dipahami dan dirawat)
3. Kestabilan (perubahan kecil dalam requirement tidak berarti perubahan yang signifikan dalam sistem yang sedang dikembangkan)
4. Lebih mudah disesuaikan dengan perubahan

Structure VS OO (Object Oriented)

1. Structured

   Pendekatan masalah berorientasi pada aksi atau data

2. Object-Oriented

   Pendekatan masalah berorientasi pada aksi dan data

Pendekatan Terstruktur	Pendekatan Objek
dikenal dengan (*Structured Analisys and Design* / SSAD)	dikenal dengan (*Object-oriented Analysis and Design* / OOAD)
Pendekatan Fungsional	Pendekatan Objek
dekomposisi permasalahan dilakukan berdasarkan fungsi atau proses secara hirarki, mulai dari konteks sampai proses-proses yang paling kecil	dekomposisi permasalahan dilakukan berdasarkan objek-objek yang ada dalam sistem
SSAD lebih sulit digunakan dalam pembangunan sistem.	OOAD lebih mudah digunakan dalam pembangunan sistem.
Pada SSAD tidak fokus pada coding	Pada OOAD lebih fokus pada coding
Pada SSAD menekankan pada kinerja team	Pada OOAD tidak menekankan pada kinerja team

Model	Analisis	Perancangan
Terstruktur	Flowchart Sistem Data flow Diagram Logis	Flowchart Program DFD Fisik STD
Objek	UML: Use Case Diagram Activity Diagram	UML: Class Diagram Activity Diagram Sequence Diagram Komponen Diagram

Gambar: Structure VS OOAD

Unified Modeling Language (UML)

Pengertian

1. UML stands for Unified Modeling Language

   Adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem

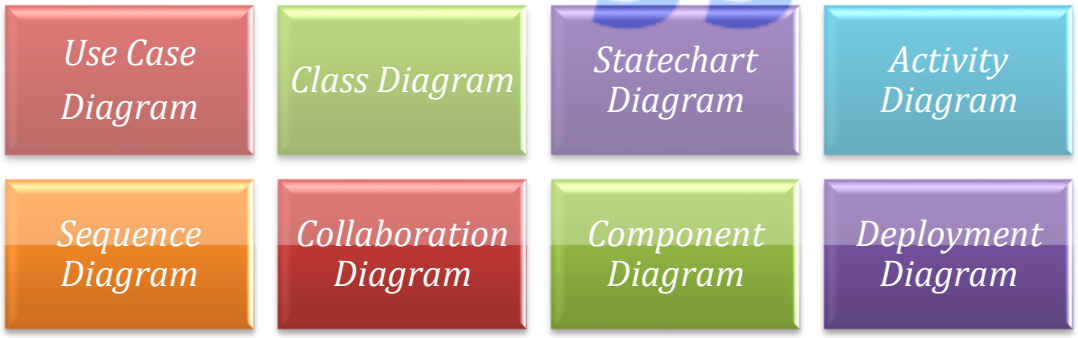
2. Merupakan kombinasi dari:
   - Data Modeling concepts (Entity Relationship Diagrams)
   - Business Modeling (Work Flow)
   - Object Modeling
   - Component Modeling

3. Supports **modeling**, **analysis** and **design** of object-oriented development systems

4. UML (Unified Modeling Language) adalah sebuah bahasa untuk menentukan, visualisasi, kontruksi, dan mendokumentasikan artifact (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. Artifact dapat berupa model, deskripsi atau perangkat lunak) dari system perangkat lunak, seperti pada pemodelan bisnis dan system non perangkat lunak lainnya.

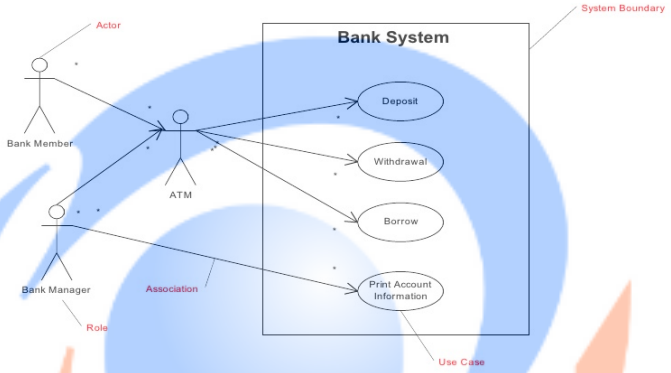
5. UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan system yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.
6. UML bukan saja merupakan bahasa visual saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemrograman, seperti JAVA, C++, Visual Basic atau bahkan dihubungkan secara langsung kedalam OODB
7. Pendokumentasiannya : requirement, arsitektur, design, source code, project plan, test dan prototype
8. Pendekatan analisa dan rancangan dengan model OO diperkenalkan sejak 1970-akhir 1980
9. Jumlah yang menggunakan metode OO mulai diuji coba dan diaplikasikan antara 1989 hingga 1994
10. OOSE *(Object Oriented Software Engineering)* oleh Grady Booch dari Rational Software Co, dan James Rumbaugh dari General Electric yang dikenal dengan OMT *(Object Modelling Language)*
11. Standarasisasi -> UML (Oktober 1994)
12. UML di standarisasi oleh OMG *(Object Management Group)*

Diagram UML



1. *Use Case Diagram*

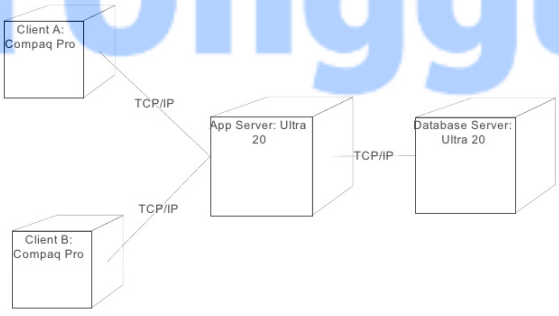
Use case adalah abstraksi dari interaksi antara system dan actor. Use case bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah system dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah system dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana system akan terlihat di mata user. Sedangkan use case diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan client.



Gambar : Use Case Diagram

2. *Deployment Diagram*

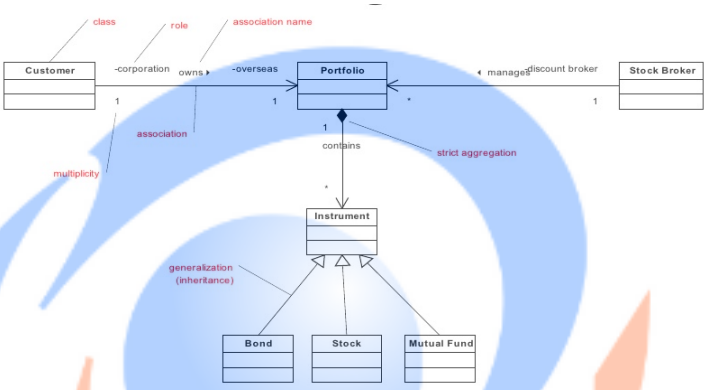
Menggambarkan tata letak sebuah system secara fisik, menampakkan bagian-bagian software yang berjalan pada bagian-bagian hardware, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes*, *executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.



Gambar: Deployment Diagram

3. *Class Diagram*

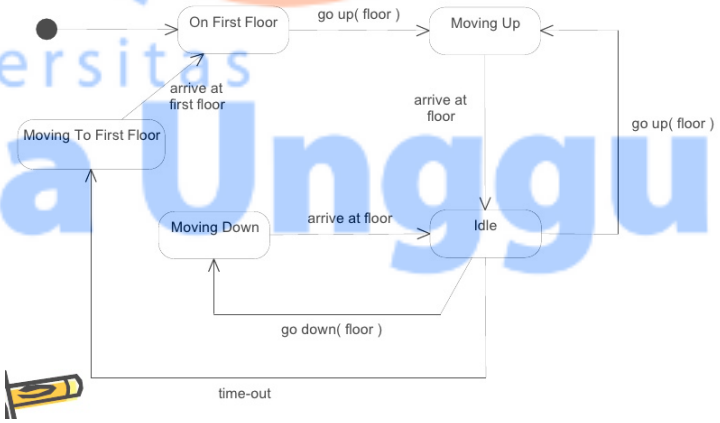
Class adalah dekripsi kelompok obyek-obyek dengan property, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya class diagram dapat memberikan pandangan global atas sebuah system. Hal tersebut tercermin dari class- class yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem biasanya mempunyai beberapa *class diagram*. Class diagram sangat membantu dalam visualisasi struktur kelas dari suatu system.



Gambar: Class Diagram

4. *Statechart Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state yang lain) suatu objek pada sistem sebagai akibat dari stimuli yang diterima.



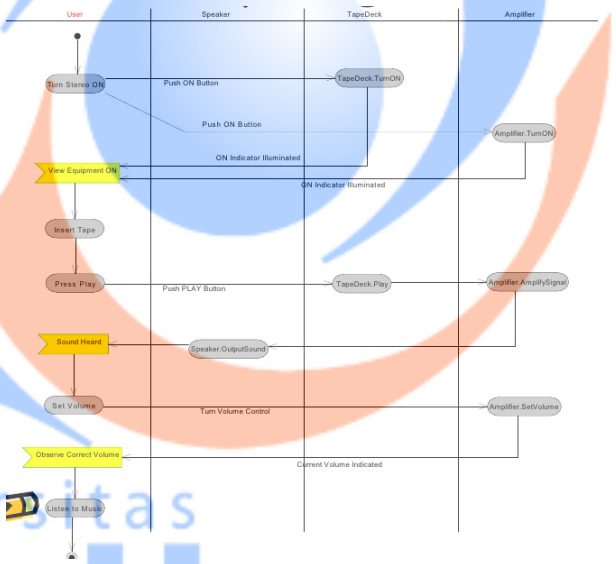
Gambar: Statechart Diagram

5. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing–masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir.

No.	Simbol	Fungsi
1	Swimlane	Menujukkan siapa yang bertanggung jawab melakukan aktivitas dalam suatu diagram.
2	Start State	Menujukkan dimana aliran kerja itu di mulai.
3	End State	Menujukkan dimana aliran kerja itu berakhir.
4	Action State	*Action state* adalah langkah-langkah dalam sebuah activity. *Action* bisa terjadi saat memasuki *activity*, meninggalkan *activity*, atau pada *event* yang spesifik.
5	Decission	Menunjukkan dimana sebuah keputusan perlu di buat dalam aliran kerja.
6	Synchronization	*Synchronization* menunjukkan dua atau lebih langkah dalam aliran kerja berjalan secara serentak.

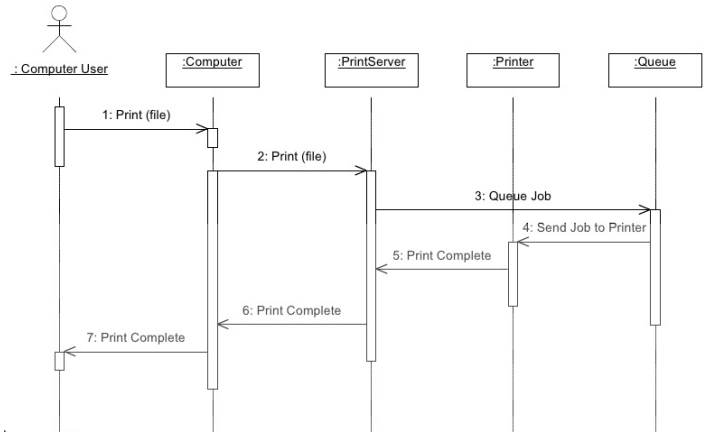
Gambar: Simbol Activity Diagram



Gambar: Contoh Activity Diagram

6. *Sequence Diagram*

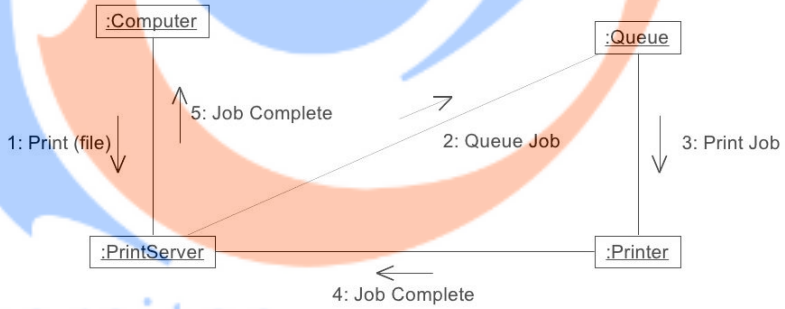
Sequence diagram menggambarkan interaksi antar objek didalam dan di sekitar sistem berupa waktu yang digambarkan terhadap waktu.



Gambar: Sequence Diagram

7. *Collaboration Diagram*

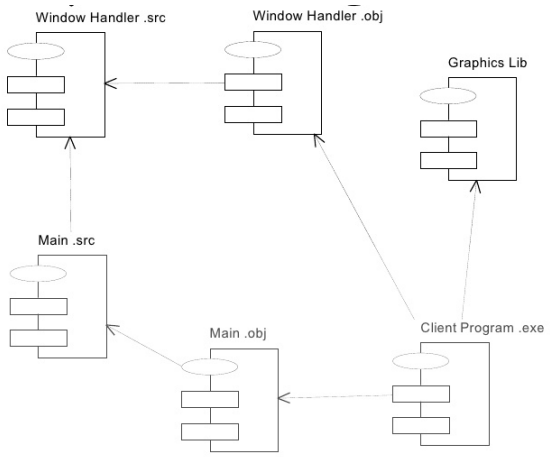
Menggambarkan kolaborasi dinamis seperti *sequence diagrams*. Dalam menunjukkan pertukaran pesan, *collaboration diagrams* menggambarkan *object* dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan *sequencediagrams,* tapi jika penekanannya pada konteks gunakan *collaboration diagram*.



Gambar: Collaboration Diagram

8. *Component Diagram*

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



Gambar: Component Diagram

Tujuan Pengenalan UML

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
2. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.
3. Memberikan model yang siap pakai, bahsa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
4. UML bisa juga berfungsi sebagai sebuah *(blue print)* cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bisa diketahui informasi secara detail tentang *coding* program atau bahkan membaca program dan menginterpretasikan kembali ke dalam bentuk diagram *(reserve enginering).*

E. Latihan

1. Jelaskan pengertian dari OOA dan OOD?
2. Jelaskan pengertian dari OOA dan OOAD?
3. Jelaskam konsep dasar dari OOAD?
4. APa yang audara ketahui perihal class dan aoject?
5. Tujuan menggunakan OOAD
6. Sebutkan perbedaan pemrogramana terstruktur dan OOAD
7. UML adalah
8. Gambarkan dan jelaskan diagram UML

9. Activit diagram adalah

10. Berikan contoh activy diagram

F. Rangkuman

OOAD adalah metode analisis yang memerikasa requirements dari sudut pandang kelas kelas dan objek yang ditemui dalam ruang lingkup permasalahan yang mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek system atau subsistem. UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat blue print atas visinya dalam bentuk yang baku. Salah satu diagram UML adalah diagram activity. Diagram ini mirip dengan flowchart, diagram activity juga bermanfaat apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan

G. Tes Formatif

1. SIMULA di perkenalkan pertama kali pada tahun …..
   a. 1950
   b. 1960
   c. 1970
   d. 1980
   e. 1990

2. Hal penting dalam pengembangan berorientasi objek adalah: …
   a. Konsep mengidentifikasi dan mengorganisasi domain aplikasi
   b. Konsep pemodelan
   c. Karakteristik objek
   d. Abstraksi
   e. Tidak Ada Yang Benar

3. Model yang menggambarkan struktur statis dari suatu objek dalam sistem dan relasinya adalah
   a. Model objek
   b. Model class
   c. Model dinamik
   d. Model fungsional

e. Tidak ada yang benar

4. Dibawah ini merupakan bentuk-bentuk objek, kecuali..
   a. classes
   b. interfaces
   c. usecases
   d. nodes
   e. deployment
5. Tiga diagram baru yang ada pada UML 2.0 adalah
   a. Composite Diagram, Class Diagram, Timing Diagram
   b. Class Diagram, Interaction Diagram, Composite Diagram
   c. Composite Diagram, Interaction Diagram, Timing Diagram
   d. Timing Diagram, Use case Diagram, Class Diagram
   e. Sequence Diagram, Class Diagram,
6. Untuk menggambarkan interaksi anatar objek dimana penekanan pada jalur menggunakan diagram
   a. communication
   b. class
   c. state machine
   d. specification
   e. activity
7. Mekanisme pembayaran model menggunakan, kecuali
   a. specification
   b. adornments
   c. common division
   d. extensibility
   e. accessibility
8. Diagram yang menggambarkan bagaimana even mengubah objek selama aktif adalah
   a. communication
   b. class
   c. state machine
   d. specification
   e. activity
9. Dibawah ini adalah diagram-diagram yang termasuk kedalam behavior diagram adalah, kecuali
   a. activity diagram
   b. interaction diagram
   c. state machine diagram
   d. class diagram
   e. use case diagram
10. Jenis relasi yang bisa timbul pada use case diagram adalah, kecuali
    a. Association antara actor dan use case
    b. Association antara use case
    c. Generalization antara actor dan use case

d. Generalization/Inheritance antara use case

e. Generalization/Inheritance antara actors

H. Daftar Pustaka

1. Alan Denis, Barbara Haley Wixon, David Tagerden, System Analys and Design : an Object – Oriented Approach with UML 2.0, John Willey and Sons, 2005
2. Edward V. Berard, Origins Objects Oriented Technology
3. Henry C. Lucas Jr., The Analysis, Design and Implementation of Information Systems 4th , McGraw Hill, 1992
4. Satzinger, Jackson, Burd, Object-Oriented Analysis and Design with the Unified Process, Course Technology, 2005
5. Simon Bennet, Steve McRobb and Ray Farmer, Object-Oriented System Analysis and Design Using UML, McGraw Hill, 2006
6. Wendy and Michael Boggs, UML with Rational Rose 2002, Sibex Inc., 2002

Universitas
Esa Unggul