4. OBJECT ORIENTED SOFTWARE ENGINEERING

15. OBJECT ORIENTED ANALYSIS, DESIGN, PROGRAMMING, AND TESTING

15.1 Paradigma Berorientasi Objek

15.2 Pendekatan Konvensional vs OO

15.3 Object Oriented Analysis

15.4 Object Oriented Design

15.5 Object Oriented Programming

15.6 Object Oriented Testing

KEMBALI

15.1 Paradigma Berorientasi Objek

2. Paradigma berbasis data, proses, dan objek

Perbedaan utama paradigma berorientasi objek dibandingkan konvensional adalah pada penyatuan proses / fungsi dan data ke dalam bentuk yang ter-enkapsulasi, sedangkan paradigma konvensional memisahkan data dengan proses.

5. Data, Attribut, Encapsulation, Method, Event

Data (sekumpulan atribut) di-enkapsulasi (dikemas / dibungkus) bersama algoritma (operasi / metode / servis) untuk melakukan suatu proses berdasarkan pesan (stimulan / message / event) yang masuk ke objek. Pesan ini mrpk sarana interface antar-objek.

3. Object, Class, Hierarchy, Inheritance

Objek-objek mrpk realitas yang unik shg tidak efisien bila dideskripsikan satu persatu sementara mereka memiliki beberapa kesamaan.

Misal kerbau dengan sapi adalah dua objek yang berbeda, ttp keduanya adalah pemakan rumput. Oleh sebab itu diciptakan kelas yang memuat karakteristik yang sama dari kerbau dan sapi. Dg dmk kelas ini dapat menginformasikan berbagai binatang pemakan rumput (ttp tidak bisa menyebutkan binatang-binatang di kebun binatang).

Kelas ini dengan mudah dikembangkan untuk merekrut objek / binatang2 lain pemakan rumput secara efisien tanpa mengulang-ulang deskripsi yang sama. Bahkan dapat dibuat hirarkhinya sebagai superkelas atau subkelas yang masing-masing memiliki warisan (inheritance) dari kelas di atasnya.

4. Polymorphism

Polymorphism diperlukan guna memperluas sistem OO yang ada secara efisien, yakni dengan nama yang sama ttp operasinya berbeda.

Misal untuk operasi penggambaran grafik yang berbeda-beda desainnya tetap sama :

graphtype draw

Hal dimungkinkan karena setiap objek memiliki operasi draw-nya sendiri.

KEMBALI

15.2 Pendekatan Konvensional vs OO

Perbedaan pendekatan konvensional dan OO tidak dikotomis sebagai dua kutub yang berseberangan atau berlawanan atau berbeda total. Keduanya memiliki beberapa dimensi pemodelan yang sama, bahkan dalam Bridge System Development Methodology keduanya dapat berfungsi secara sinergis maupun komplementer.

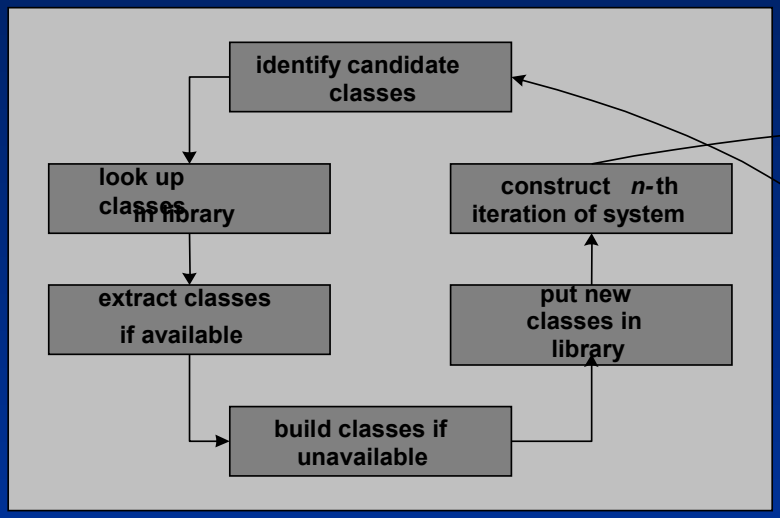
Perbedaan yang menyolok terletak pada dimensi Dekomposisi Top-down dan Urutan Pemrosesan End-to-end yang selalu hadir dalam analisis terstruktur tetapi jarang ada pada OOA. Keseluruhan dimensi tsb sbb.

15.2.1 Dimensi pemodelan

1. Identifikasi / klasifikasi entitas
2. Umum ke spesifik dan keseluruhan ke hub entitas bagian
3. Hubungan dg entitas lain
4. Gambaran atribut entitas
5. Partisi model skala besar
6. Keadaan dan transisi antarkeadaan
7. Spesifikasi detil untuk fungsi
8. Dekomposisi top-down
9. Urutan pemrosesan end-to-end
10. Identifikasi pelayanan eksklusif
11. Komunikasi entitas (melalui pesan)

15.2.2 OO Process Model

Component Assembly Process Model (Bab 3) dpt digunakan untuk menggambarkan OO Process Model (OO Analysis, OO Design, OO Programming, OO Testing) dg memperlakukan ‘komponen’ sbg ‘objek’ sbb.



Engineering,
contruction & release

customer evaluation

entry point

Customer communication

risk analysis

planning

KEMBALI

15.3 Object Oriented Analysis

15.3.1 Identifikasi Kelas dan Objek

Proses ini diawali dengan menguji problem statement (rumusan masalah) atau dengan penguraian gramatikal pada narasi pemrosesan pada sistem yang akan dikembangkan. Objek ditentukan dengan menggarisbawahi kata benda atau klausa benda serta memasukkannya pada tabel untuk dianalisis.

Objek dapat berupa :

5. Entitas eksternal (mis sistem lain, perangkat, manusia) yang menggunakan atau menghasilkan informasi.
6. Hal / benda (mis laporan, tampilan, sinyal) yang mrpk bag domain informasi.
7. Kejadian / event (mis transfer properti atau pelengkapan serial gerakan robot) yang terjadi dalam konteks operasi sistem.
8. Peran (mis manajer, perekayasa, penjual) yang dilakukan oleh orang yang berinteraksi dg sistem.
9. Unit-unit organisasi (mis divisi, kelompok, tim)
10. Tempat (mis manufacturing floor, loading dock) yang membangun konteks masalah dan keseluruhan fungsi sistem.
11. Struktur (mis kendaraan roda empat atau komputer) yang menentukan kelas dari objek, atau scr ekstrim kelas objek yang sesuai.

a. Contoh identifikasi objek pd PL Safehome

Objek / Kelas Potensial	Klasifikasi
Pemilik rumah	Peran / entitas eksternal
Sensor	Entitas eksternal
Kontrol panel	Entitas eksternal
Instalasi	Kejadian
Sistem (sistem keamanan)	Benda
Nomor, tipe	Bukan objek ttp atribut sensor
Master password	Benda
Nomor telpon	Benda
Event sensor	Kejadian
Alarm audible	Entitas eksternal
Monitoring service	Unit organisasi / Entitas eksternal

b. Karakteristik pengujian objek

1. Informasi yang disimpan : bila informasi mengenai objek hrs diingat
2. Pelayanan yang diperlukan : objek hrs memiliki serangkaian operasi untuk mengubah nilai atribut
3. Atribut bertingkat : fokus harus berupa informasi mayor, bukan objek dengan atribut tunggal
4. Atribut umum : serangkaian atribut dapat didefinisikan untuk objek tsb
5. Operasi umum : serangkaian operasi dapat diterapkan pada objek tsb dan sesuai dg kejadian objek
6. Persyaratan dasar : berupa entitas eksternal yang menghasilkan atau mengkonsumsi informasi.

c. Hasil Identifikasi

Objek / Kelas Potensial	Hasil
Pemilik rumah	Ditolak : nomor 1,2 gagal sekalipun nomor 6 OK
Sensor	Diterima : semua diterapkan
Kontrol panel	Diterima : semua diterapkan
Instalasi	Ditolak
Sistem (sistem keamanan)	Diterima : semua diterapkan
Nomor, tipe	Ditolak : 3 gagal, atribut sensor
Master password	Ditolak : 3 gagal
Nomor telpon	Ditolak : 3 gagal
Event sensor	Diterima : semua diterapkan
Alarm audible	Diterima :2,3,4,5,6 diterapkan
Monitoring service	Ditolak : 1,2 gagal bahkan sekalipun 6 diterapkan

15.3.2 Pemodelan kelas

1. Tipe dan Karakteristik Kelas

Tipe Kelas

- Device Class : sensor, motor, keyboard, dsb.
- Property Class : rating kredit, gaji pokok, dsb.
- Interaction Class : memodelkan interaksi antarobjek, mis pembelian, peminjaman, dsb.

Karakteristik Kelas

- Tangibility : kenampakan fisik atau abstrak
- Inclusiveness: atomic atau agregat (ada objek bersarang)
- Sequentiality: konkaren (ada urutan kontrol sendiri) atau sekuensial (kontrol luar).
- Persistence : transient (timbul / tenggelam saat operasi), temporer (timbul selama operasi dan mati saat operasi berakhir), permanen (basis data).
- Integrity : mengijinkan korupsi (tidak melindungi sumber dayanya dari pengaruh luar) atau melindungi sumber daya (melalui kontrol).

2. CRC Model

- Guna memudahkan identifikasi kelas/ objek dan hub dapat digunakan Class – Responsibility – Collaborator Model (CRC Model). Responsibility menunjukkan atribut dan operasi yang dilakukan kelas, Collaborator adalah kelas – kelas yang diperlukan untuk menyelesaikan responsibility, melalui permintaan informasi / aksi.

Nama Kelas :	
Tipe Kelas : (mis perangkat, properti, peran, event, dsb)	
Karakteristik Kelas : (mis tangible, atomik, temporer, dsb)	
Tanggung Jawab	Kelas-kelas Kolaborator

15.3.3 Metode OOA

Banyak metode yang dikembangkan untuk OOA seperti Booch, Coad dan Yourdon, Jacobson, Rambaugh, Wirfs – Brock. Outline dari metode Booch sbb.

1. Identifikasi kelas dan objek
2. Identifikasi semantik (tanggung jawab, operasi, dan kolaborasi) dari kelas dan objek
3. Identifikasi hub (ketergantungan) antar-kelas (atau objek)
4. Penyaringan (hirarkhi dan pengklasteran kelas)
5. Implementasi kelas dan objek

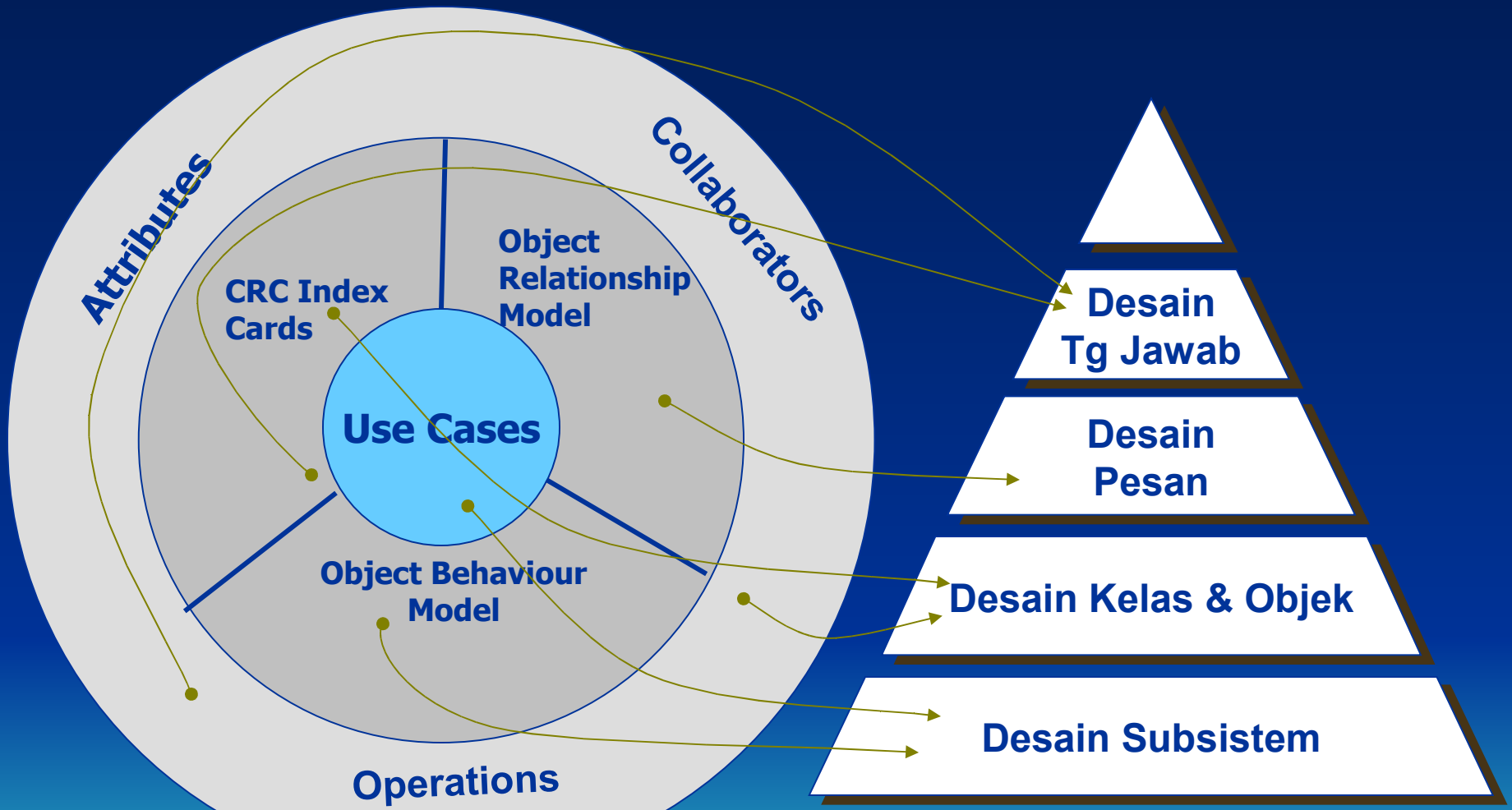
KEMBALI

15.4 Object Oriented Design

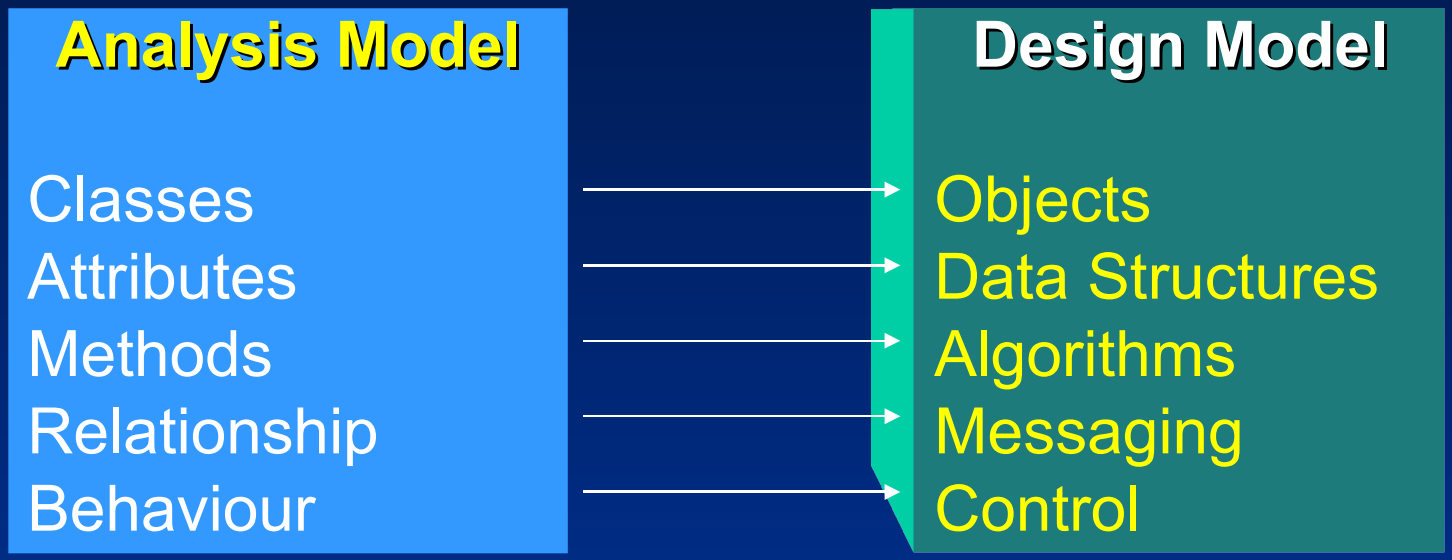
OOD memiliki kemampuan lebih dibanding metode konvensional dalam menerapkan 4 konsep desain PL seperti abstraksi, information hiding, independensi fungsional, dan modularitas. OOD mengaplikasikan desain data (bila ada atribut), desain interface (bila ada proses pemesanan), dan desain operasional. OOD membentuk 'modul-modul' yang lebih nyata.

Dibandingkan dg pendekatan konvensional arsitektur OOD tidak memperlihatkan hirarkhi kontrol ttp kolaborasi antarobjek dengan aliran kontrol.

15.4.1 Translasi Model OOA ke Model OOD



15.4.1 Translasi Model OOA ke Model OOD (lanj)



15.4.2 Metode OOD

Seperti metode OOA banyak peneliti memberikan gagasannya. Berikut ini OOD menurut Booch :

- Perencanaan Arsitektur : pengklasteran, pelapisan objek sesuai tingkat abstraksi, identifikasi skenario proses, validasi prototype desain.
- Desain Taktis : penentuan aturan penggunaan operasi dan atribut, dan aturan manajemen, penanganan kesalahan, dan fungsi infrastruktur lainnya; pengembangan skenario dan prototype; serta pengkajian.
- Perencanaan Rilis : pengumpulan skenario yang dikembangkan selama OOA, perancangan dan pembangunan rilis arsitektur scr inkremental, penyesuaian tujuan dan jadwal rilis inkremental sesuai kebutuhan.

KEMBALI

15.5 Object Oriented Programming

OOP memperluas model desain ke dalam domain yang dapat dieksekusi mesin. Kelas, atribut, operasi, dan pesan diterjemahkan dalam bentuk machine executable.

Pada prinsipnya alat penterjemah (bahasa pemrograman) apapun dapat digunakan, yang membedakan adalah tingkat kesulitan yang akan dihadapi pemrogram. Setiap bahasa pemrograman memiliki sintaks, semantik, tatabahasa, dan pustaka komponen-komponen yang dapat memudahkan pemrogram dalam menyusun perintah-perintah untuk mesin.

Bahasa C++ merupakan salah satu alat yang disediakan untuk OOP, selain kompatibilitasnya dengan versi sebelumnya yang disediakan untuk pemrograman prosedural. ADA, Smalltalk, Basic, Pascal dan beberapa bahasa lain juga kini disediakan untuk OOP. Faktor penentu berada di tangan perekayasa khususnya pemrogram sehubungan dengan penggunaan bahasa tersebut sesuai dengan prinsip-prinsip object oriented ataukah tidak.

KEMBALI

15.6 Object Oriented Testing

Prinsip-prinsip pengujian OOT hampir sama dengan metode pengujian konvensional yang sudah dibahas sebelumnya.

2. Pengujian Kelas : analog dengan pengujian unit. Bila pengujian konvensional berfokus pada detil algoritma, OOT untuk kelas berfokus pada pengendalian operasi dan perilaku kelas.
3. Pengujian Terintegrasi : pada OOT tidak menggunakan top-down atau bottom-up karena tidak adanya struktur kontrol yang hirarkhis shg pendekatan integrasi inkremental tidak mungkin dilakukan. Strategi dalam OOT :
    - Pengujian Thread-based : mengintegrasikan himpunan kelas yang dibutuhkan untuk merespon input melalui pengintegrasian thread.
    - Pengujian Use-based : merupakan pengujian terhadap kelas-kelas independen yang kemudian dilanjutkan dengan kelas-kelas dependen.
4. Pengujian Validasi : merupakan pengujian tingkat sistem yang berfokus pada aksi yang dapat dilihat oleh user. Metode pengujian black box dapat digunakan untuk mengendalikan pengujian ini.

Untuk melaksanakan OOT tetap diperlukan Test Case seperti pada pengujian konvensional.

***

KEMBALI