

Analisis & Perancangan Berorientasi Objek

Metode Coad - Yourdon

oleh

Ir. I Gede Made Karma, MT

Latar Belakang

Termasuk metode yang muncul saat awal metode berorientasi objek mulai berkembang
⇒ metode yang masih *naif* (terlalu sederhana)

Pengaruh metode analisis & perancangan tradisional masih kental

Sejalan dengan waktu, metode ini terus dikembangkan sehingga mengadopsi konsep-konsep yang muncul belakangan. (di kuliah ini masih diberikan metode aslinya)

Coad & Yourdon menyatakan 7 motivasi kunci dan keuntungan analisis & perancangan berorientasi objek dibandingkan metode analisis tradisional:

- Menangani domain persoalan yang makin menantang
- Meningkatkan interaksi antara analis and ahli pada domain persoalan
- Meningkatkan konsistensi internal antara analisis, perancangan, dan pemrograman
- Secara eksplisit menyatakan kesamaan antara kelas & objek
- Membuat spesifikasi yang lebih tangguh terhadap perubahan
- Mengguna-ulang hasil OOA, OOD dan OOP
- Menyediakan representasi yang konsisten antara analisis, perancangan dan pemrograman

Latar Belakang (2)

Menurut Coad & Yourdon hasil utama OOA/OOD:

mengurangi kompleksitas persoalan dan tanggung jawab sistem di dalamnya.

Metode OOA/OOD didasarkan atas sejumlah prinsip umum *mengatasi kompleksitas* yang tidak dimiliki oleh metode lain (misalnya: Dekomposisi Fungsional, Data Flow, Information Modelling - basis dari E-R):

- 1) Abstraksi
 - a) Prosedural
 - b) Data
- 2) Enkapsulasi
- 3) Inheritance/Pewarisan
- 4) Asosiasi
- 5) Komunikasi dengan pesan
- 6) Sebaran cara organisasi
 - a) Objek & Atribut
 - b) Whole & parts
 - c) Kelas & anggotanya, serta perbedaan di antara mereka
- 7) Skala
- 8) Kategori kelakuan
 - a) Penyebab langsung
 - b) Perubahan sejalan waktu
 - c) Kesamaan fungsi

Latar Belakang (3)

Menurut Coad & Yourdon, ancangan OO didasarkan atas representasi yang seragam antara Kelas & Objek.

Implikasinya:

- Tidak ada perbedaan besar antara notasi analisis & perancangan
- Tidak ada *transisi* dari analisis ke perancangan
- Tidak ada model *waterfall* yang harus diikuti; model *spiral* dan *incremental* juga bisa diterapkan
- Ada sejumlah keterampilan dan strategi yang diperlukan oleh analis & perancang
- Adanya keseragaman representasi dari OOA ke OOD ke OOP

Konsep dan Konstruksi

Konsep:

- **Objek:** adalah abstraksi dari sebuah entitas nyata/tidak nyata (tangible or intangible) yang informasinya harus diingat/disimpan: nilai-nilai atribut dan layanan-layanan eksklusif dienkapsulasi
- **Kelas:** adalah deskripsi dari satu atau lebih objek dengan sejumlah atribut dan layanan yang seragam termasuk deskripsi tentang cara membuat objek dari kelas tersebut
- **Atribut:** adalah sejumlah data (informasi keadaan) di mana tiap objek dari suatu kelas mempunyai nilai tersendiri
- **Class-&-Object:** adalah suatu istilah yang berarti "suatu kelas dan objek-objek yang ada pada kelas tersebut "

Konsep dan Konstruksi (2)

- **Subjek:** adalah mekanisme untuk membagi model yang besar dan kompleks. Subjek juga berguna untuk mengatur paket kerja pada proyek-proyek besar berdasarkan hasil penyelidikan awal dengan OOA.
- **Layanan (*Service*):** adalah kelakuan spesifik yang dilakukan oleh objek yang menjadi tanggung jawab objek tersebut
- **Keadaan/Status (*State*):** Status dari sebuah objek adalah gabungan dari nilai-nilai atribut objek tersebut
- **Transisi (*Transition*):** adalah perubahan status
- **Kondisi (*Condition*):** aksi *If-precondition-trigger-terminate*
- **Text Block:** Teks
- **Loop:** aksi *While-Do-Repeat-trigger-terminate*

Konsep dan Konstruksi (3)

Hubungan antar-objek

- Struktur **Whole-Part**

Satu objek (yang mewakili **Whole**) dapat didekomposisi menjadi objek-objek lain (**Parts**).

Ada 3 variasi struktur whole-part:

- 1) *Assembly-Parts* : contoh: sebuah **mobil** mempunyai **roda, mesin, chasis...**
- 2) *Container-Contents* : contoh: sebuah **kotak** berisi sejumlah **paku...**
- 3) *Collection-Members* : contoh: sebuah **organisasi** mempunyai sejumlah **analisis, manajer...**

Hubungan Whole-Part relationships dapat memiliki rentang spesifik, seperti layaknya konsep kardinalitas pada pemodelan E-R

- **Instance connections** (Hubungan asosiasi)

Yaitu koneksi yang diperlukan oleh sebuah objek dengan objek lain dalam rangka memenuhi tanggung jawabnya.

Contoh: Sebuah objek **Orang** bekerja pada objek **Kantor**

Juga bisa memiliki rentang

Konsep dan Konstruksi (4)

- **Message Connections**

Memodelkan kebergantungan pemrosesan sebuah objek yang dinyatakan dengan kebutuhan atas layanan-layanan dari objek lain dalam rangka memenuhi tanggung jawabnya (layanan yang disediakan olehnya) sendiri.

Contoh: Untuk memenuhi tanggung jawab (layanan) "*permohonan cuti*", objek **Orang** perlu layanan "*persetujuan cuti*" dari objek **Manajer**

Hubungan antar-kelas

- Struktur **Generalization-Specialization/Gen-Spec** (Hubungan pewarisan)

Mendefinisikan hirarki pewarisan untuk kelas-kelas yang merupakan spesialisasi dari kelas lain yang lebih umum (general).

Sebuah kelas bisa mewarisi sifat dari sebuah *superclass* (kelas general) yang disebut dengan pewarisan tunggal (*single inheritance*) atau dari sejumlah superclass yang disebut dengan pewarisan ganda (*multiple inheritance*).

Contoh: seorang **Manajer** adalah spesialisasi dari **Orang**.

Konsep dan Konstruksi (5)

Operasi dan komunikasi

Jenis layanan yang disediakan pada OOA adalah:

- *Algorithmically simple services* seperti Create, Connect, Access and Release
- *Algorithmically complex services*:
 - Layanan Menghitung (*Calculation*) yang menghitung hasil komputasi dari nilai-nilai sebuah objek.
 - Layanan Pemantauan (*Monitoring*) yang memantau sistem atau piranti eksternal

Pada metode OOA objek berkomunikasi dengan cara *mengirim pesan* (permintaan layanan).

Objek pengirim mengirimkan pesan yang diterima oleh objek penerima untuk kemudian oleh objek penerima tersebut dilakukan sejumlah aksi dan mengembalikan hasil ke objek pengirim pesan.

Mekanisme ini pada OOA diagram dinyatakan dengan *Message Connections*.

Ancangan Umum

Aktivitas utama **analisis berorientasi objek** (OOA) adalah:

1. Mencari/menemukan kelas & objek (*Finding Class & Objects*)
2. Mengidentifikasi struktur (*Identifying Structures*)
3. Mengidentifikasi (*Identifying Subjects*)
4. Mendefinisikan atribut (*Defining Attributes*)
5. Mendefinisikan layanan (*Defining Services*)

Langkah-langkah ini *bukan langkah sekuensial*

Model analisis yang dihasilkan dari aktivitas analisis berorientasi objek metode Coad & Yourdon adalah 5 lapisan:

1. Lapisan subjek (*A Subject layer*)
2. Lapisan kelas & objek (*A Class & object layer*)
3. Lapisan struktur (*A Structure layer*)
4. Lapisan atribut (*An Attribute layer*)
5. Lapisan layanan (*A Service layer*)

Ancangan Umum

Pada saat **perancangan berorientasi objek** (OOD), kelima lapisan yang dihasilkan oleh analisis berorientasi objek ditambahkan dengan 4 komponen lain yang harus dirancang untuk masing-masing lapisan:

1. Human Interaction Component (HIC)
2. Problem Domain Component (PDC)
3. Task Management Component (TMC)
4. Data Management Component (DMC)

Keempat komponen ini juga *bukan langkah sekuensial*

Keberadaan keempat komponen ini juga *tidak wajib ada*

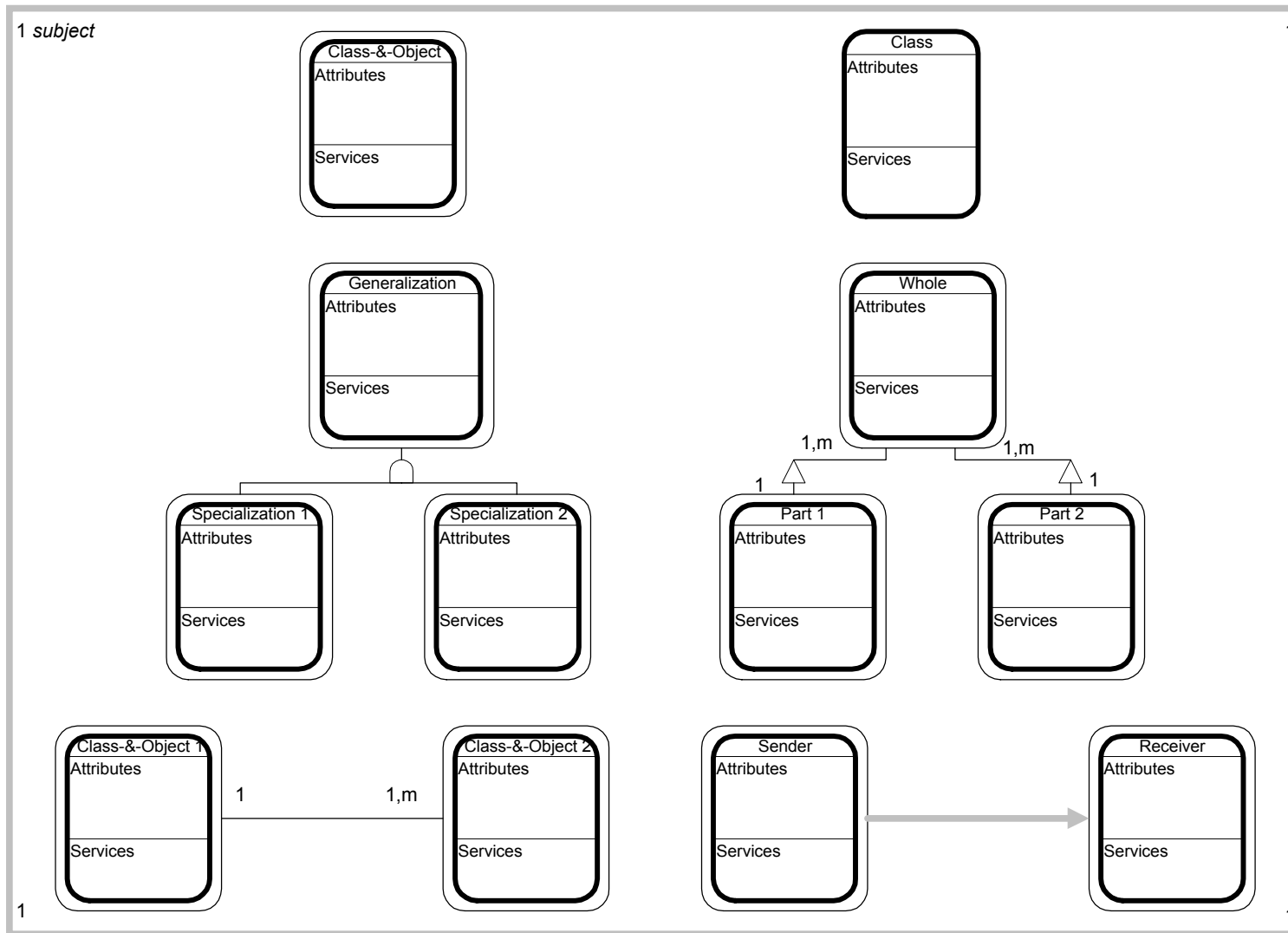
Teknik-Teknik yang Digunakan

- Hasil dari penerapan OOA/OOD adalah satu OOA diagram yang terdiri dari 5 lapisan:
 - 1) *Subject layer* sebagai mekanisme partisi
 - 2) *Class & Object layer* untuk menangkap Kelas dan Objek
 - 3) *Structure layer* untuk menangkap struktur pewarisan dan whole-part
 - 4) *Attribute layer* untuk menangkap atribut dan instance connections antara Kelas dan Objek
 - 5) *Service layer* untuk menangkap metode objek (layanan) dan message connections antara Kelas dan Objek
- Kelakuan dinamik dari sebuah kelas ditangkap di *Object State Diagrams*, yang merupakan bentuk terbatas dari State Transition Diagrams.
- Algoritma yang diterapkan untuk satu layanan dinyatakan dengan *Service Charts*, yang merupakan satu jenis *flowcharts*.
- Hubungan antara sebuah layanan dari Service Chart dan keadaan/status dari Object State Diagram dinyatakan dengan *Service/State Table* (satu layanan terdefinisi untuk state tertentu).

Catatan:

Object State Diagrams, Service Charts dan Service/State Tables tidak dijelaskan dengan baik pada buku OOA/OOD

Notasi Grafis



Proses Analisis (Aktivitas 1) - (1)

Definisi analisis:

- Memisahkan atau menguraikan suatu keseluruhan menjadi bagian-bagiannya untuk mendapatkan sifat, proporsi, fungsi, relasi, dsb dari bagian-bagian tersebut.
- Praktik mempelajari ranah persoalan untuk mendapatkan spesifikasi dari kelakuan eksternal yang dapat diamati; pernyataan kebutuhan yang lengkap, konsisten dan layak; lingkup karakteristik fungsional dan operasional yang terkuantifikasi (misalnya keandalan, ketersediaan, kinerja)

Sebelum melakukan analisis maupun perancangan berorientasi objek, perlu dilakukan pemeriksaan hasil analisis berorientasi objek sebelumnya pada ranah persoalan yang sama/serupa \Rightarrow guna ulang (*reuse* - isu utama pada ancangan berorientasi objek)

Langkah-langkah berikut (aktivitas x.x) tidak perlu dilakukan secara sekuensial

Untuk sistem besar, ada baiknya penguraian subjek dilakukan sebelum mengidentifikasi class-&-object

Proses Analisis (Aktivitas 1.1) - (1)

Mengidentifikasi Kelas & Objek (Aktivitas 1.1)

Aktivitas 1.1.1.: mempelajari ranah persoalan

⇒ akuisisi informasi

Aktivitas 1.1.2: mencari Kelas & Objek potensial/kandidat, dengan melihat:

- Struktur
- Sistem lain
- Piranti
- Benda-benda/Kejadian-kejadian yang harus diingat
- Peran yang dibawakan
- Prosedur operasional
- Situs (Lokasi Fisik)
- Unit Organisasional

Proses Analisis (Aktivitas 1.1) - (2)

Aktivitas 1.1.3: Kelas & Objek potensial yang ditemukan diberi nama

Aktivitas 1.1.4: Kelas & Objek potensial diuji untuk dijadikan Kelas & Objek akhir menggunakan kriteria:

- Perlu diingat
- Kelakuan yang diperlukan
- (Biasanya dinyatakan dengan) banyak atribut
- (Biasanya dinyatakan dengan / menghasilkan) lebih dari satu objek (instansiasi) pada suatu kelas
- Atribut yang selalu terpakai
- Layanan yang selalu terpakai
- Kebutuhan dasar/normal dari ranah persoalan
- Bukan merupakan hasil turunan

Aktivitas 1.1.5: Menambahkan Kelas & Objek pada diagram analisis berorientasi objek (OOA diagram)

Proses Analisis (Aktivitas 1.2) - (1)

Mengidentifikasi Struktur (Aktivitas 1.2)

Aktivitas 1.2.1: Mencari struktur Gen-Spec

Tiap kelas dipertimbangkan sebagai Generalization, lalu uji dengan bertanya:

1. Apakah kelas masih dalam ranah persoalan?
2. Apakah kelas masih dalam tanggung jawab persoalan?
3. Apakah akan ada turunan/pewarisan?
4. Apakah spesialisasi/generalisasi masih memenuhi kriteria sebagai kelas & objek?

Lalu tiap kelas dianggap sebagai Specialization, lalu uji lagi dengan pertanyaan-pertanyaan di atas

Proses Analisis (Aktivitas 1.2) - (2)

Aktivitas 1.2.2: Mencari struktur Whole-Part

Cari kemungkinan hubungan yang bisa terjadi:

- Assembly-Parts
- Container-contents
- Collection-members

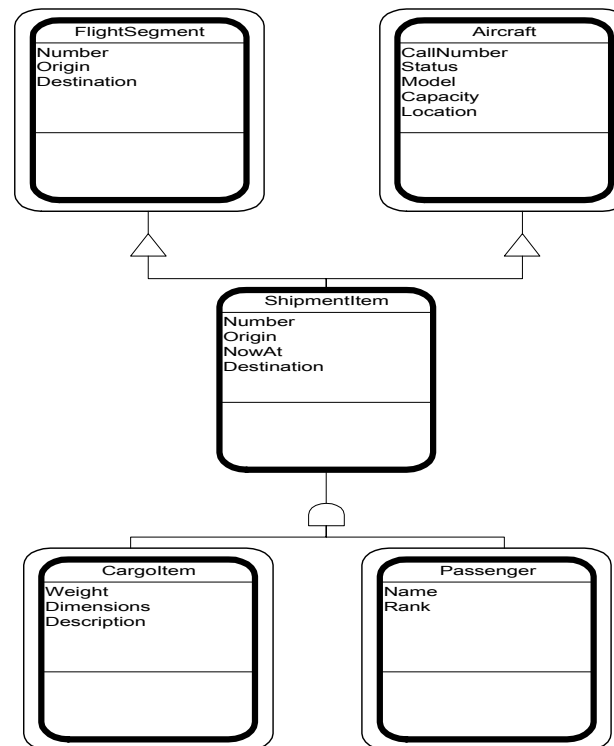
Tiap objek dipertimbangkan sebagai Whole dan sebagai Part, lalu uji dengan pertanyaan:

1. Apakah kelas/objek masih dalam ranah persoalan?
2. Apakah kelas/objek masih dalam tanggung jawab persoalan?
3. Apakah kelas/objek menangkap lebih dari sekedar nilai status?
4. Jika tidak, masukkan atribut sebagai Whole
5. Apakah kelas/objek memberikan abstraksi yang berguna?

Proses Analisis (Aktivitas 1.2) - (3)

Aktivitas 1.2.3: Identifikasikan struktur berganda (*multiple structures*)

Struktur berganda adalah berbagai kombinasi Gen-Spec, Whole-Part, atau bahkan kombinasi antara keduanya



Aktivitas 1.2.4: Menambahkan struktur-struktur yang telah teridentifikasi kepada diagram analisis berorientasi objek (diagram OOA)

Proses Analisis (Aktivitas 1.3) - (1)

Mengidentifikasi Subjek (Aktivitas 1.3)

Aktivitas 1.3.1: Memilih subjek-subjek yang mungkin ada dengan:

- Menjadikan kelas teratas pada tiap struktur yang telah teridentifikasi sebagai Subjek
- Menjadikan kelas & objek yang tidak menjadi anggota struktur sebagai Subjek

Aktivitas 1.3.2: Menghaluskan subjek

Dilakukan dengan mencari ketergantungan (*interdependencies*) minimal dan interaksi minimal antara Kelas & Objek pada subjek-subjek yang berbeda.

Aktivitas 1.3.3: Mengkonstruksi subjek

Dilakukan dengan

- Pada lapisan subjek menggambarkan kotak subjek, nama subjek, dan nomor subjek dan bila perlu daftar kelas & objek yang ada pada subjek.
- Pada lapisan lain gambarkan subjek dan kotak-kotak subjek berlabel

Aktivitas 1.3.4: Menambahkan subjek-subjek (atau mengubah susunan atak - *layout*) yang ada pada diagram analisis berorientasi objek (diagram OOA)

Proses Analisis (Aktivitas 1.4) - (1)

Mengidentifikasi atribut (Aktivitas 1.4)

Aktivitas 1.4.1: Mencari atribut dengan menerapkan pertanyaan berikut pada tiap kelas/objek:

1. Bagaimana saya (sebuah kelas/objek) digambarkan secara umum?
2. Bagaimana saya (sebuah kelas/objek) digambarkan pada ranah persoalan ini?
3. Bagaimana saya (sebuah kelas/objek) digambarkan pada konteks tanggung jawab sistem ini?
4. Apa yang perlu saya (sebuah kelas/objek) ketahui?
5. Apa informasi yang perlu saya ingat sepanjang waktu?
6. Pada kondisi (*state*) apa sajakah saya bisa berada?

Aktivitas 1.4.2: Menempatkan atribut setinggi mungkin pada hirarki pewarisan pada struktur Gen-Spec

Proses Analisis (Aktivitas 1.4) - (2)

Aktivitas 1.4.3: Mengidentifikasi *Instance Connections* antar objek
Dilakukan dengan menambah garis koneksi untuk tiap objek yang menyatakan pemetaan di dalam ranah persoalan.
Penempatan Instance Connection harus hati-hati terutama yang berhubungan dengan struktur Gen-Spec, harus diperhatikan pada kelas mana sebetulnya koneksi terjadi.

Aktivitas 1.4.4: Pengujian atribut dan *Instance Connections* untuk beberapa kasus tertentu.

Pengujian kasus khusus untuk atribut:

- Atribut yang tidak memiliki nilai terpakai \Rightarrow kemungkinan struktur Gen-Spec tambahan
- Kelas & Objek yang hanya memiliki satu atribut \Rightarrow kemungkinan bagian/atribut dari Kelas & Objek lain
- Atribut yang memiliki nilai berulang \Rightarrow kemungkinan membentuk Kelas & Objek lain

Proses Analisis (Aktivitas 1.4) - (3)

Pengujian kasus khusus untuk *Instance Connections*:

- *Many-to-many Instance Connections* \Rightarrow kemungkinan ada atribut pada koneksi sehingga membentuk Kelas & Objek baru (kejadian yang harus diingat)
- *Instance connections* antar objek-objek yang berasal dari satu Kelas \Rightarrow kemungkinan ada atribut pada koneksi sehingga membentuk Kelas & Objek baru (kejadian yang harus diingat)
- Instance Connections berganda antar objek \Rightarrow kemungkinan ada atribut pada koneksi sehingga membentuk Kelas & Objek baru (kejadian yang harus diingat)
- Kebutuhan akan *Instance Connection* tambahan akibat pembentukan Kelas & Objek baru hasil pengujian di atas (bila antar kedua Kelas-&-Objek semula tetap ada koneksi yang tidak perlu melibatkan kelas-&-objek baru yang terbentuk)
- Objek satu (dari satu-ke-banyak) yang terkoneksi punya arti khusus \Rightarrow bila perlu menambahkan atribut pada objek banyak

Aktivitas 1.4.5: Menambahkan atribut dan *Instance Connections* pada diagram analisis berorientasi objek.

Proses Analisis (Aktivitas 1.5) - (1)

Mengidentifikasi Layanan (Aktivitas 1.5)

Aktivitas 1.5.1: Mengidentifikasi *state-state* yang mungkin dari Objek Dilakukan dengan menggambarkan state serta transisi yang mungkin terjadi dengan **Object State Diagram**.

Aktivitas 1.5.2: Mengidentifikasi layanan yang dibutuhkan untuk tiap Kelas-&-Objek Untuk layanan yang bersifat *algorithmically complex services* seperti Calculate & Monitor, perlu digambarkan **Service Chart** untuk memperlihatkan kelakuan algoritmik dari layanan

Jika diperlukan **Service/State Tables** dapat digambarkan untuk memperlihatkan koneksi antara layanan dan state

Layanan sederhana tidak dituliskan eksplisit pada diagram analisis berorientasi objek.

Proses Analisis (Aktivitas 1.5) - (2)

Aktivitas 1.5.3: Mengidentifikasi *Message Connections*

Dilakukan dengan menerapkan pertanyaan berikut pada tiap objek::

1. Dari objek lain manakah saya (kelas-&-objek) memerlukan layanan?
2. Objek lain manakah yang memerlukan layanan dari saya (kelas-&-objek)?

Lalu telusuri tiap *message connections* dan ulangi pertanyaan di atas.

Aktivitas 1.5.4.: Menambahkan layanan dan *message connections* pada diagram analisis berorientasi objek (diagram OOA)

Proses Analisis (Aktivitas 1.6) - (1)

Siapkan dokumentasi (Aktivitas 1.6)

Aktivitas ini terdiri dari:

- **Aktivitas 1.6.1:** Menggambarkan diagram analisis berorientasi objek yang lengkap
- **Aktivitas 1.6.2:** Menuliskan spesifikasi Kelas-&-Objek
- **Aktivitas 1.6.3:** Tambahkan dokumentasi suplemental jika diperlukan yang terdiri dari:
 - Tabel jalur eksekusi kritis (*critical threads of execution*)
 - Kendala-kendala sistem tambahan
 - Tabel atau diagram *Services/State*

Proses Perancangan (Aktivitas 2)

Definisi perancangan:

- Pembuatan rencana asli, sketsa, pola, dan lain-lain
- Praktik menggunakan spesifikasi dari kelakuan eksternal yang teramati dan menambahkan detail-detail yang diperlukan untuk implementasi sistem komputer sesungguhnya, termasuk detail interaksi manusia, manajemen *task*, dan manajemen data.

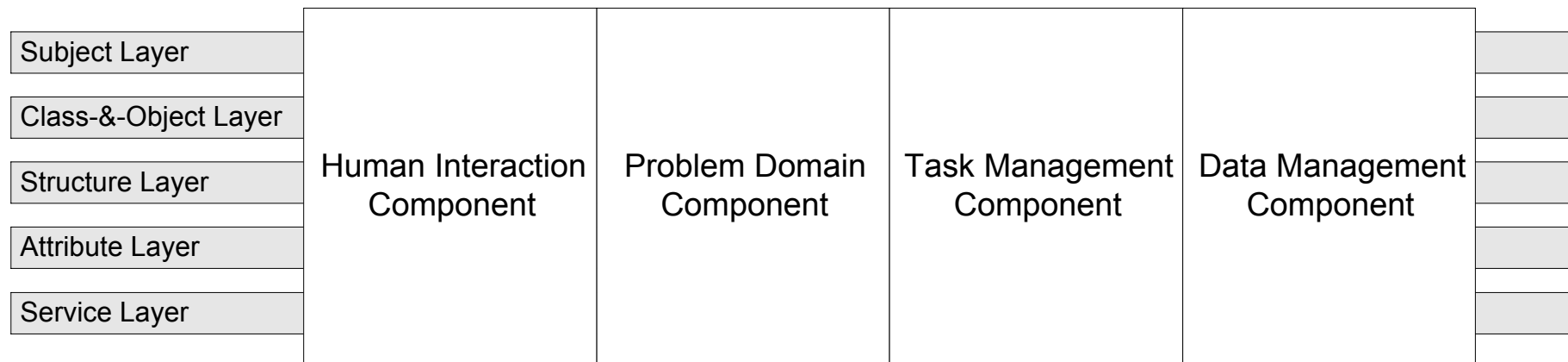
Tujuan (objektif) dari perancangan:

- Meningkatkan produktivitas \Rightarrow mengurangi waktu pengujian & penghilangan cacat, meningkatkan guna ulang
- Meningkatkan kualitas \Rightarrow ketiadaan cacat, kemudahan penggunaan, *portability*, kemudahan modifikasi
- Mempertinggi tingkat keremawatan (*maintainability*) \Rightarrow

Model Lengkap

Terdiri dari 5 lapisan dan 4 komponen (4 aktivitas yang membentuk):

Komponen:	Aktivitas:
1. Human Interaction Component (HIC)	1. Perancangan HIC
2. Problem Domain Component (PDC)	2. Perancangan PDC
3. Task Management Component (TMC)	3. Perancangan TMC
4. Data Management Component (DMC)	4. Perancangan DMC



Proses Perancangan (Aktivitas 2.1) - (1)

Aktivitas 2.1: Merancang *Problem Domain Component*

Pada perancangan berorientasi objek, hasil dari analisis berorientasi objek diteruskan ke tahap perancangan dengan memperbaiki dan menambahkan pada hasil yang ada.

Aktivitas 2.1.1: Mencari rancangan dan kelas-kelas yang ada untuk diguna-ulang pada hasil analisis berorientasi objek.

Rancangan atau kelas-kelas yang ada bisa berasal dari proyek terdahulu atau menggunakan *off-the-shelf classes*.

Bila perlu struktur Gen-Spec digunakan untuk menerapkan OTS-classes ke dalam hasil analisis berorientasi objek yang ada.

Aktivitas 2.1.2: Pengelompokan kelas-kelas ranah persoalan yang spesifik.

Dilakukan dengan menambah Kelas Akar sebagai mekanisme pengelompokan. Cara ini adalah jalan tengah bagi pertentangan antara *understandibility* dan isi semantik dari model.

Proses Perancangan (Aktivitas 2.1) - (2)

Aktivitas 2.1.3: Memantapkan protokol antar objek dengan menambahkan kelas Generalisasi.

Protokol yang dimaksudkan adalah penamaan dari sejumlah Layanan yang sama pada sekumpulan kelas.

Aktivitas 2.1.4: Mengakomodasikan aras pewarisan yang didukung oleh bahasa pemrograman.

Pada sejumlah bahasa pemrograman berorientasi objek, pewarisan tidak diperbolehkan atau dibatasi (satu aras saja). Pada kasus ini, struktur Gen-Spec harus diuraikan/disesuaikan agar dapat diimplementasikan pada bahasa yang dituju.

Jenis pewarisan berganda:

- Wajik Sempit (*Narrow Diamond*)
- Wajik Lebar (*Wide Diamond*)

Proses Perancangan (Aktivitas 2.1) - (3)

Ancangan pemetaan struktur pewarisan ganda ke struktur pewarisan tunggal:

- Pemecahan ke hirarki ganda dengan pemetaan di antaranya. Pemetaan dengan struktur Whole-Part atau Instance Connection
- Pemampatan menjadi hirarki tunggal. Sejumlah struktur Gen-Spec tidak lagi eksplisit

⇒ tidak dianjurkan pada perancangan karena semantik dari pewarisan

Aktivitas 2.1.5: Meningkatkan Kinerja.

Kinerja yang dimaksudkan adalah kecepatan (*speed*)

Kecepatan ditingkatkan dengan mengukur *coupling* antar kelas (lalu lintas pemanggilan pesan)

Kecepatan relatif (yang dirasakan) ditingkatkan dengan menyimpan data/hasil perhitungan sementara.

Proses Perancangan (Aktivitas 2.1) - (4)

Aktivitas 2.1.6: Mendukung *Data Management Component*

Dilakukan dengan:

- Menambah kemampuan tiap objek untuk menyimpan dirinya sendiri
- Kemampuan menyimpan dilakukan oleh Data Management Component.
Untuk kedua ancangan tersebut atribut & layanan khusus perlu ditambahkan ke PDC
- Menggunakan Object Oriented Database Management System (OODBMS) untuk melakukan penyimpanan objek

Aktivitas 2.1.7: Menambahkan komponen-komponen aras rendah

Untuk kenyamanan implementasi (rancangan & pemrograman), komponen-komponen aras rendah diisolasi (dikumpulkan) pada kelas-kelas terpisah

Aktivitas 2.1.8: Mengkaji ulang (*review*) dan menguji penambahan pada hasil analisis berorientasi objek.

Hasil perancangan PDC harus sesuai dengan semantik hasil analisis berorientasi objek
Usahakan perubahan sesedikit mungkin terjadi.

Proses Perancangan (Aktivitas 2.2) - (1)

Aktivitas 2.2: Merancang Human Interaction Component (HIC)

Teknik pembuatan purwarupa dianjurkan.

Aktivitas 2.2.1: Mengklasifikasikan orang yang menjadi pengguna

Dilakukan dengan permainan peran. Klasifikasi pengguna dilakukan dengan kriteria:

- Tingkat keterampilan (*skill level*)
- Tingkat organisasional
- Keanggotaan pada kelompok (contoh: staf atau pelanggan)

Aktivitas 2.2.2: Tiap kategori orang diberi penjelasan termasuk *task scenario*.

Yang harus dituliskan:

- Siapakah kategori orang tersebut
- Tujuan kategori orang
- Karakteristik (seperti: umur, pendidikan dll)
- *Critical success factors (needs/wants dan likes/dislikes/biases)*
- Tingkat keterampilan (*Skill level*)
- *Task Scenarios*

Proses Perancangan (Aktivitas 2.2) - (2)

Aktivitas 2.2.3: Merancang hirarki perintah

Dilakukan dengan mempelajari sistem interaksi manusia yang ada sekarang

Untuk menghaluskan/menyempurnakan hirarki tersebut, panduan tersebut harus dipertimbangkan:

- Urutan layanan
- Pembagian (*chunking*) berdasarkan pola whole-part
- Pembagian (*chunking*) berdasarkan breadth/depth (sebaiknya dalam dan lebar tidak lebih dari 3
 - mengikuti saran Miller tentang kemampuan optimal manusia dalam mengingat benda pada suatu saat: 7 ± 2)
- Meminimalkan jumlah langkah (*click, point, dsb*) untuk mendapat layanan lengkap

Proses Perancangan (Aktivitas 2.2) - (4)

Aktivitas 2.2.4: Merancang interaksi detail

Kriteria untuk mengkaji kesuksesan detail interaksi:

- Konsistensi pada interaksi manusia
- Meminimalkan jumlah langkah
- Menyediakan umpan balik yang berarti bagi pengguna
- Gunakan langkah sederhana untuk menyelesaikan satu tugas \Rightarrow memberikan pengguna rasa *menguasai*
- Sediakan fungsi '*Undo*'
- Tidak mengandalkan ingatan manusia untuk mengingat segala sesuatu
- Menjamin waktu belajar dan usaha yang minimal
- Kepuasan dan daya tarik sistem pada pengguna adalah penting

Proses Perancangan (Aktivitas 2.2) - (5)

Aktivitas 2.2.5: Membuat purwarupa

Purwarupa digunakan untuk menguji kriteria-kriteria yang dipatok pada aktivitas sebelumnya.

Aktivitas 2.2.6: Mendefinisikan/merancang kelas-kelas *Human Interaction Component*

Rancangan kelas-kelas HIC sangat bergantung kepada lingkungan implementasinya (pemrogramannya). Kelas-kelas HIC antara lain: windows, fields, graphics dan selectors.

Aktivitas 2.2.7: Sedapat mungkin *graphical user interfaces* (GUI) standar digunakan. Tingkat standarisasi bergantung pada lingkungan yang ingin digunakan.

Proses Perancangan (Aktivitas 2.3) - (1)

Aktivitas 2.3: Merancang *Task Management Component*

Task = nama lain dari proses

Proses = alur (*stream*) aktivitas yang didefinisikan oleh kode sumbernya

Aktivitas 2.3.1: Menentukan apakah pendefinisian task (*multitasking*) diperlukan oleh sistem

Contoh sistem yang perlu task:

- Sistem dengan akuisisi data dan tanggung jawab kontrol untuk piranti lokal
- Interaksi manusia-mesin dengan banyak jendela yang berjalan simultan
- Sistem banyak pengguna (*multiuser*)
- Arsitektur perangkat lunak *multi-subsystem*
- Sistem pada pemroses tunggal yang perlu task untuk saling berkomunikasi & berkoordinasi
- Sistem pada banyak pemroses (*multiprocessor*)
- Sistem yang perlu berkomunikasi dengan sistem lain

Jika tidak diperlukan sebaiknya perancangan task diabaikan karena akan meningkatkan kompleksitas.

Proses Perancangan (Aktivitas 2.3) - (2)

Aktivitas 2.3.2: Mengidentifikasi jenis task

1. *Event-Driven Tasks* yang dipicu oleh sebuah event (misal: *mouse-click*)
2. *Clock-Driven Tasks* yang dipicu oleh interval waktu tertentu
3. *Priority Tasks*
4. *Critical Tasks*

Jika diperlukan tiga atau lebih task, maka dianjurkan untuk menambah sebuah task koordinator

Aktivitas 2.3.3: Menguji kebutuhan task

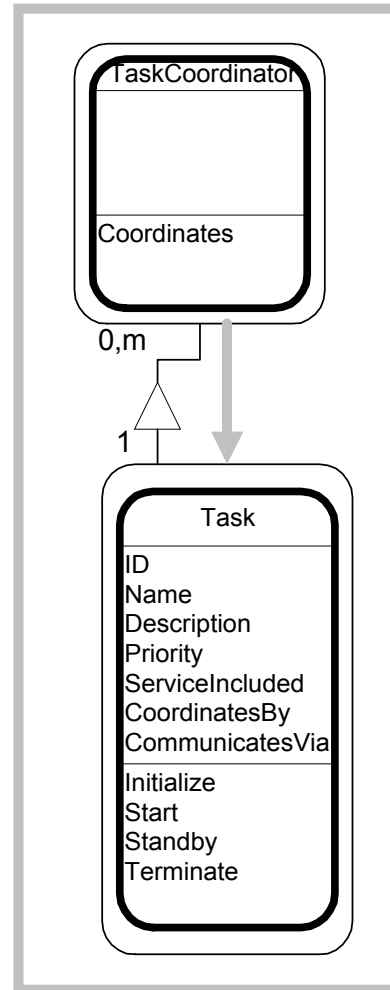
Task yang sudah terdefinisi harus diuji dengan sejumlah kriteria untuk meminimalkan jumlah task yang dipakai dan menjaga *understandability* dari tiap task

Aktivitas 2.3.4: Mendefinisikan tiap task dengan menentukan:

- Apa maksud task
- Bagaimana task dikoordinasikan/mengkoordinasikan dirinya
- Bagaimana task berkomunikasi

Proses Perancangan (Aktivitas 2.3) - (3)

Template dari task & koordinator task pada OOD metode Coad Yourdon:



Proses Perancangan (Aktivitas 2.4) - (1)

Aktivitas 2.4: Merancang *Data Management Component*

Aktivitas 2.4.1: Memilih ancangan untuk DMC

Ancangan yang dipilih antara lain salah satu dari: *Flat Files*, *Relational Database Management System* (RDBMS) atau *Object Oriented Database Management System* (OODBMS)

Berdasarkan ancangan yang dipilih, aspek-aspek lainnya juga harus dipertimbangkan misalnya: identifikasi, normalisasi.

Aktivitas 2.4.2: Memilih kakas manajemen data

Untuk ancangan yang dipilih, kakas yang akan dipakai dikaji berdasarkan sejumlah kriteria manajemen data, seperti: Concurrency, Manajemen transaksi, dll

Aktivitas 2.4.3: Merancang komponen manajemen data

Untuk ancangan dan kakas yang dipilih, rancangan detilnya dihasilkan, menyangkut atak (*layout*) data dan layanan-layanan yang diperlukan.

Strategi perancangan bergantung pada ancangan yang dipilih.

Kriteria Perancangan yang Baik (1)

Coad and Yourdon memberikan sejumlah panduan untuk perancangan berorientasi objek dalam mengukur kualitas rancangan. Panduan ini meliputi topik-topik berikut:

- ***Coupling***

= tingkat ke-saling-bergantung-an antar elemen pada rancangan

Jenis:

- *Interaction Coupling*: coupling antar objek karena hubungan *Message Connections*
- *Inheritance Coupling*: coupling antar kelas karena hubungan pewarisan:
 - Menolak warisan
 - Mengabaikan warisan

Kriteria Perancangan yang Baik (2)

- ***Cohesion***

= tingkat kontribusi suatu elemen dari sebuah bagian rancangan dalam menjalankan satu tujuan yang terdefinisi

Jenis:

- *Service Cohesion*: sebuah layanan seharusnya hanya menjalankan satu dan hanya satu fungsi
- *Class Cohesion*: sebuah kelas seharusnya hanya menyatakan satu tanggung jawab yang harus diemban oleh objeknya
- *Gen-Spec Cohesion*: sebuah kelas warisan seharusnya secara semantik merupakan spesialisasi dari kelas generalisasinya, dan bukan sekedar karena persamaan atribut/layanan.

Kriteria Perancangan yang Baik (3)

- *Reuse*

Jenis:

- Guna ulang kode
 - *Cut&Paste source code*
 - *Includes* pada tingkat kode sumber
 - Pewarisan
 - Tautan biner (*binary link*)
 - Pemanggilan saat eksekusi
- Guna ulang rancangan
- Guna ulang hasil analisis

Akan sukses dilakukan bila disertai komitmen dari organisasi untuk melakukannya

Kriteria Perancangan yang Baik (4)

Selain kriteria utama (coupling, cohesion, dan reuse), beberapa kriteria umum tambahan yang harus dipertimbangkan:

- Kejelasan rancangan
- Kedalaman *Generalization-Specialization*
- Kesederhanaan Kelas & Objek
- Kesederhanaan protokol
- Kesederhanaan layanan
- Kerawanan rancangan terhadap perubahan
- Ukuran sistem

Untuk mengevaluasi rancangan, teknik *Walk-throughs* dan penentuan *Critical Success Factors* rancangan dianjurkan oleh Coad & Yourdon.

Transformasi Perancangan ke Implementasi (1)

Coad & Yourdon memberikan sedikit arahan tentang implementasi perancangan ke bahasa pemrograman (baik yang murni berorientasi objek atau tidak)

Faktor utama dalam pemilihan bahasa pemrograman bagi hasil OOA & OOD:

Tingkat kemampuan bahasa dalam menangkap semantik dari ranah persoalan

Ini disebabkan untuk mempertahankan:

- Representasi dasar yang sama antara OOA, OOD, dan OOP
- Reusability
- Maintainability

Untuk mengevaluasi dukungan bahasa dalam konsep berorientasi objek, kriteria yang digunakan adalah dukungan atas:

- Kelas
- Objek
- Pewarisan (Generalization-Specification)
- Enkapsulasi (Whole-Part)
- Atribut
- Layanan

Transformasi Perancangan ke Implementasi (2)

Dukungan atas **kelas** = bahasa mampu mendefinisikan kelas sebagai enkapsulasi status & layanan

Dukungan atas **objek** = bahasa mampu menyatakan sejumlah objek berbeda sebagai instansiasi dari suatu kelas

Dukungan atas **pewarisan** = bahasa mampu menyatakan pewarisan dan resolusi konflik nama antara kelas orang tua & kelas anak, sedapat mungkin mendukung pewarisan ganda

Dukungan atas **enkapsulasi** = bahasa mampu menyatakan suatu kelas sebagai bentukan dari kelas lainnya

Dukungan atas **atribut** = bahasa mampu mendukung *Instance Connection* sebagai bagian dari state objek, *visibility*, dan *constraint*-nya.

Transformasi Perancangan ke Implementasi (3)

Dukungan atas **layanan** = bahasa mampu mendukung *Message Connection*, *visibility*, dan *dynamic binding*

Dukungan tambahan: dukungan atas **Object Persistence** = bahasa mampu secara terintegrasi melakukan manajemen data, baik terdefinisi langsung oleh bahasanya maupun melalui tambahan (*extension*) bahasa

Sejumlah bahasa yang dikaji oleh Coad & Yourdon (1991) atas dukungannya terhadap 6 elemen berorientasi objek:

- Bahasa berorientasi objek murni:
 - C++ dan Object Pascal
 - Smalltalk dan Objective C
 - Eiffel
- Bahasa berorientasi objek tidak murni:
 - Ada
 - C (dan bahasa prosedural lainnya)