2 - Analisis Terstruktur

Tujuan Analisis

Memodelkan suatu problem atau masalah agar lebih mudah dipahami dan dapat ditemukan solusinya.

Pemodelan pada Tahap Analisis:

* DFD – pemodelan fungsional
* ERD – pemodelan data
* STD – pemodelan behaviour

Keterkaitan antar Model Analisis:

* Data store (DFD) vs entitas / relasi (ERD)
* Proses (DFD) vs aksi (STD)

DFD – Data Flow Diagram

Pengertian DFD

Diagram yang menggambarkan proses aliran data input / output dari sebuah sistem informasi yang dibangun.

Yang Dimodelkan:

* Proses dan aliran data antar proses.
* Proses pada DFD level 1 berkaitan dengan kebutuhan fungsionalitas perangkat lunak.

Elemen:

* Entitas eksternal
* Proses
* Data flow
* Data store

ERD – Entity Relationship Diagram

Pengertian ERD

Diagram yang menjelaskan hubungan antar entitas (tabel) di dalam sebuah database.

Yang dimodelkan

* Data yang harus dikelola perangkat lunak dan relasinya.

Elemen

* Entitas

Sebuah obyek yang dapat dibedakan dengan objek lain, penamaan berupa kata benda.

Contoh: mahasiswa, bangunan, buku, rekening.

* Relasi

Hubungan antara 2 atau lebih entitas, penamaan berupa kata kerja.

Contoh melakukan, mengambil, membayar.

* Atribut

Properti yang dimiliki setiap entitas yang akan disimpan datanya, penamaan berupa kata benda.

Contoh: nama, alamat, no ktp.

* Kardinalitas

Angka yang menunjukkan jumlah suatu objek yang berhubungan dengan objek lain didalam suatu relasi.

Contoh: 1:1 (one to one), 1:M (one to many), M:M (many to many).

* Modalitas

Partisipasi sebuah entitas pada suatu relasi.

Contoh: 0 jika partisipasi bersifat optional / parsial, 1 jika partisipasi bersifat wajib / total. Contoh partisipasi total: setiap anak memiliki ibu, contoh partisipasi parsial: setiap perempuan tidak wajib memiliki anak.

STD – State Transition Diagram

Pengertian STD

Diagram yang memodelkan tingkah laku keluar-masuknya aliran data melalui sebuah sistem informasi.

Yang Dimodelkan

* Aspek dinamis perangkat lunak.

Elemen

* State
* Event
* Action

3 - Perancangan Terstruktur

Tujuan Perancangan

Memodelkan solusi agar siap diimplementasikan.

Pemodelan pada Tahap Perancangan

* Perancangan Arsitektural:

Struktur Modul

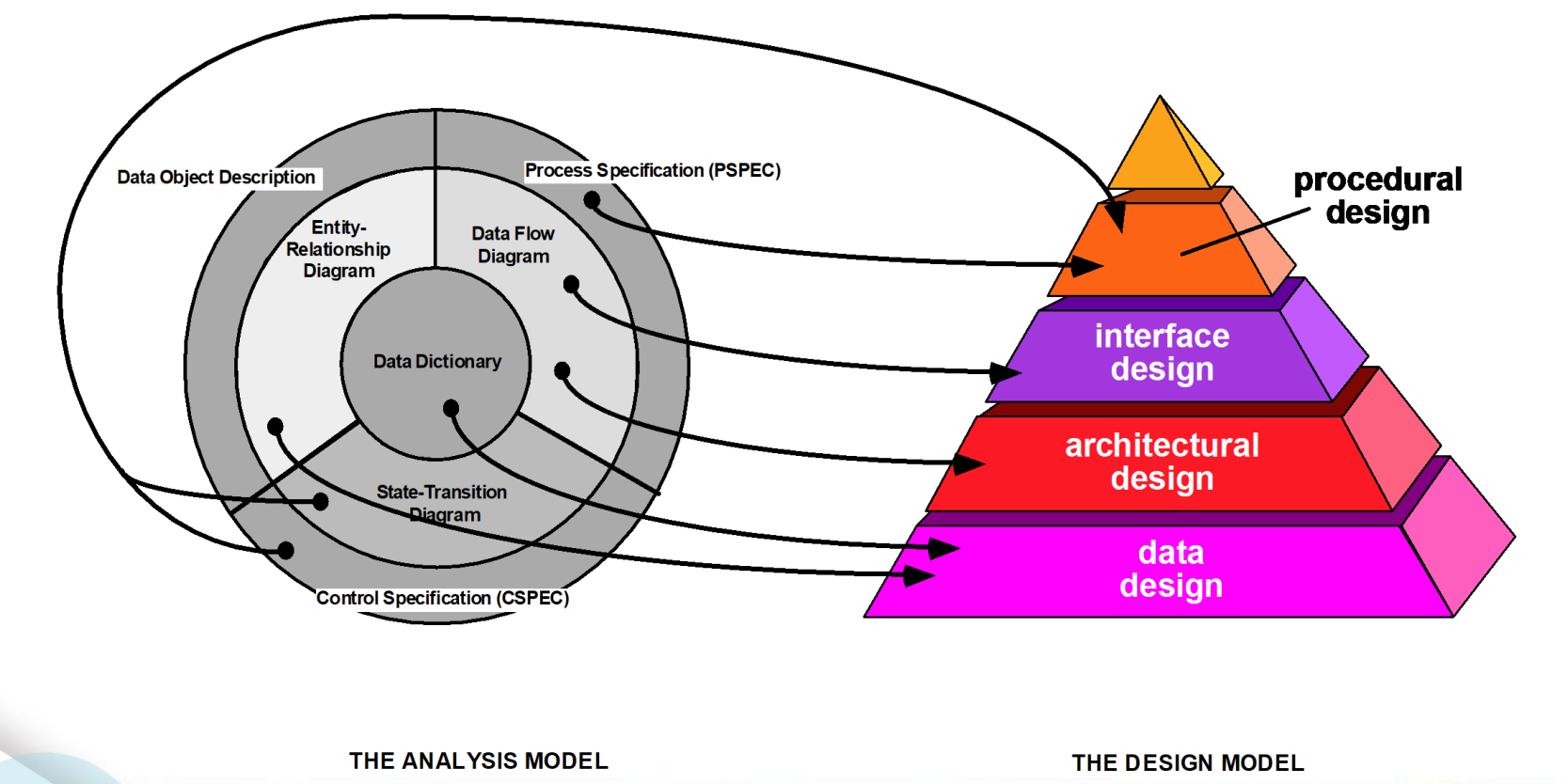
* Perancangan Antarmuka:
* Antarmuka antar-modul
* Antarmuka dengan S/W lain atau H/W
* Antarmuka dengan pengguna
* Perancangan Data:

Struktur Data dan Skema Basis Data

* Perancangan Prosedural / Component - Level:

Algoritma

Analysis to Design



Prinsip Design

* Mempertimbangkan pendekatan alternatif.
* Dirujuk dari model analisis.
* Meminimalkan jarak intelektual.
* Menampakkan keseragaman dan integrasi.
* Terstruktur untuk mengakomodasi perubahan.
* Terstruktur untuk mendegradasi lembut.
* Direview untuk meminimalkan konseptual (semantik) error.

Elemen Struktur Modul

* Modul
* Keterkaitan antar modul

Arsitektur

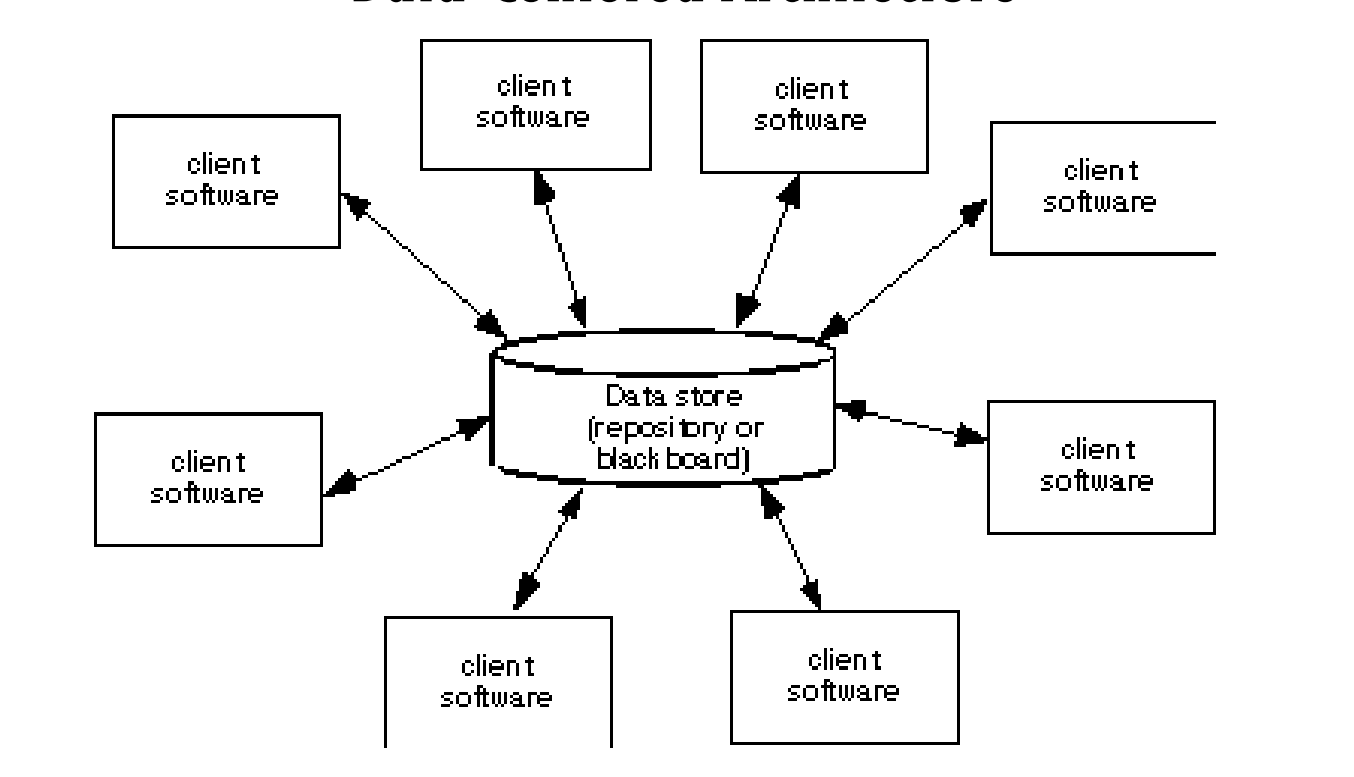
Pengertian Arsitektur

Arsitektur bukan operasi perangkat lunak. Tetapi adalah representasi yang memungkinkan seorang insinyur perangkat lunak untuk:

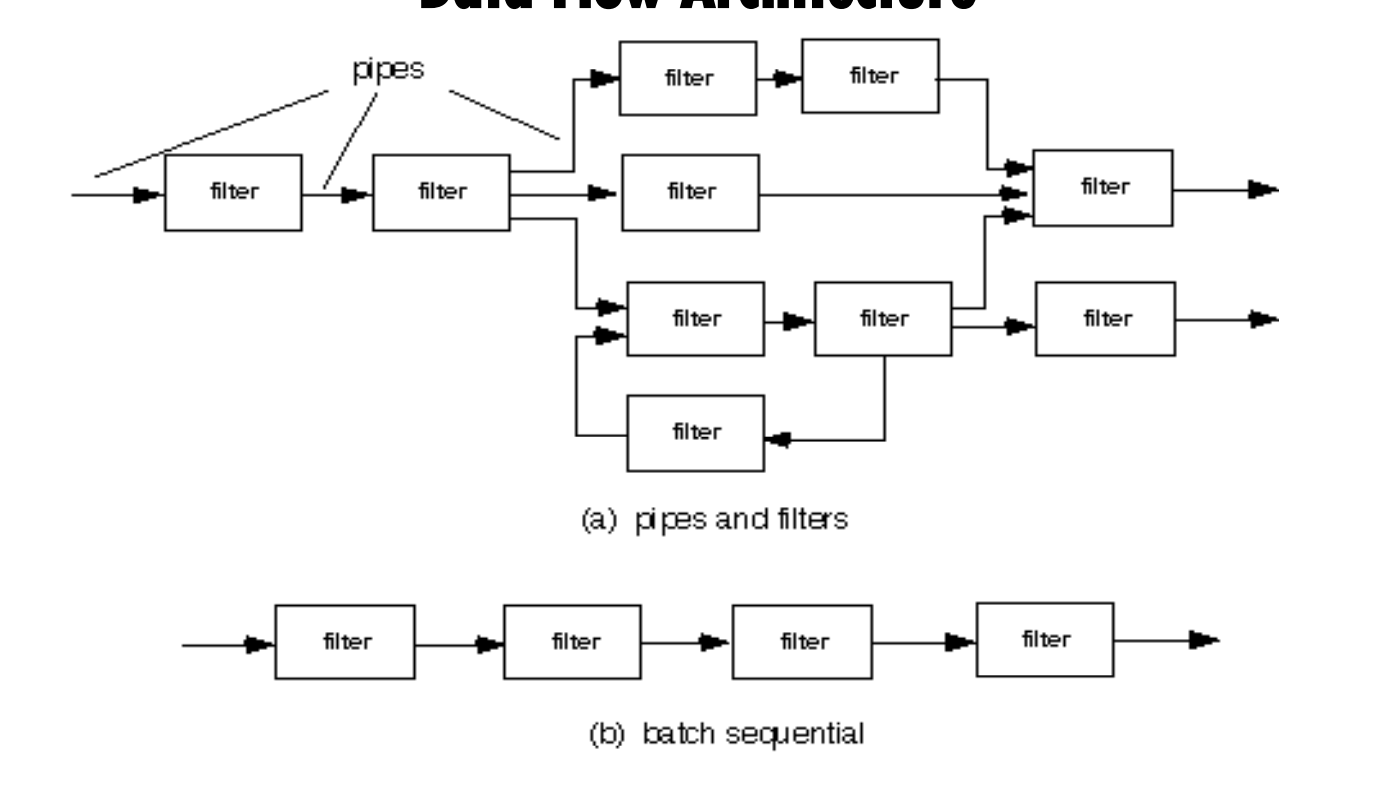
* Menganalisis efektivitas desain dalam memenuhi persyaratan yang dinyatakannya,
* Mempertimbangkan alternatif arsitektur pada tahap ketika membuat perubahan desain masih relatif mudah.
* Mengurangi risiko yang terkait dengan pembangunan perangkat lunak.

Gaya Arsitektur

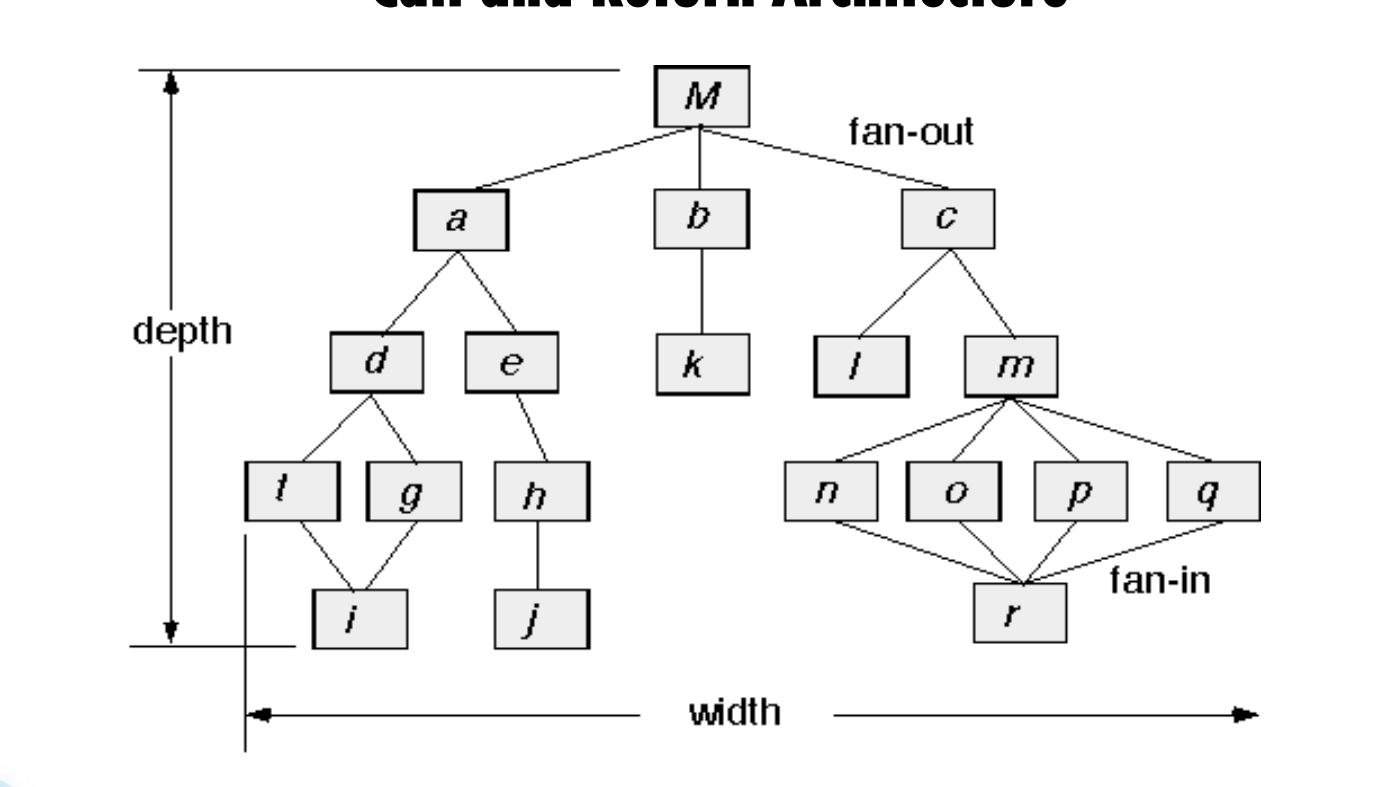
* Data-centered architectures



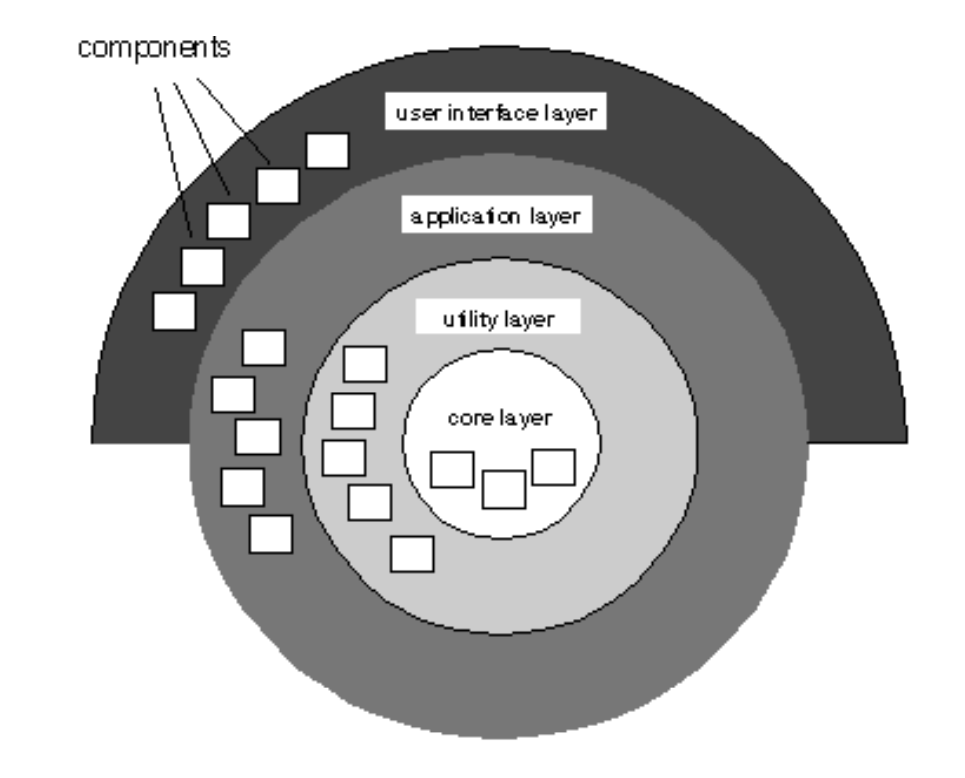
* Data flow architectures



* Call and return architectures



* Layered architectures

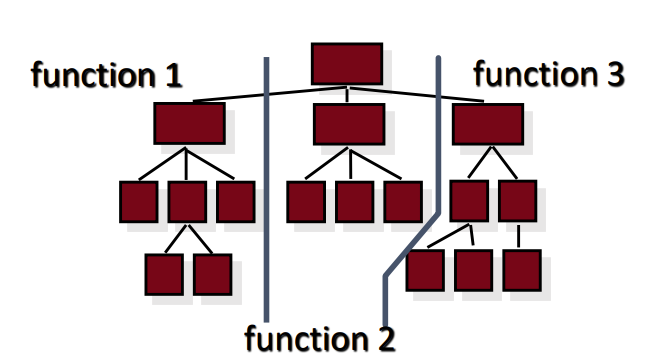


* Object-oriented architectures

Mempartisi Arsitektur

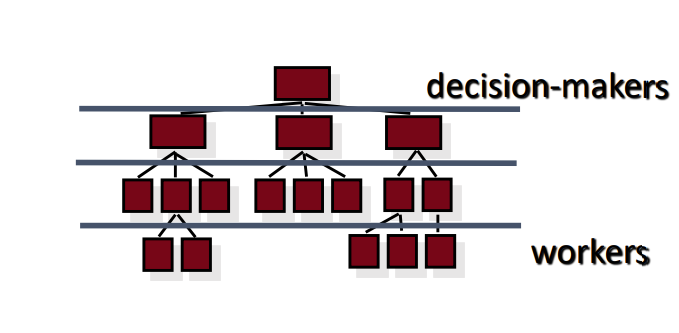
* Partisi horizontal

Mendefinisikan cabang terpisah dari hirarki modul untuk setiap fungsi utama. Menggunakan modul kontrol untuk mengkoordinasikan komunikasi antara fungsi.



* Partisi vertical

Pengambilan keputusan dan kerja yang bertingkat. Pengambilan keputusan modul harus berada di atas arsitektur.



Tujuan Mempartisi Arsitektur

* Menghasilkan software yang lebih mudah diuji.
* Menunjuk pada software yang lebih mudah dipelihara.
* Menghasilkan propagasi dari sedikit efek samping.
* Menghasilkan software yang lebih mudah diperluas.

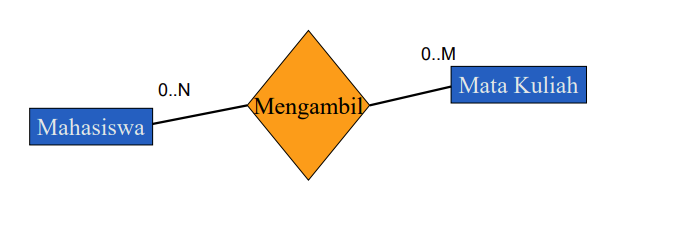
Perancangan Data

Hasil Perancangan Data

* Struktur data
* Skema basis data
* Rancangan detail setiap tabel (nama, deskripsi, volume, field, key , dll)

Tahapan Merancang Data

* Review ERD



* Memetakan menjadi skema basis data
* Tabel Mahasiswa

NIM, Nama, Alamat

* Tabel Mata Kuliah

KD\_Kul, Nama\_Kul, SKS

* Tabel Pengambilan\_Kuliah

Sem, Thn, NIM, Kd\_Kul, Nilai

Perancangan Antarmuka

Hasil Perancangan Antarmuka

* Inter-modular interface design
* External interface design
* Human-computer interface design

Arahan Merancang Antarmuka – Aturan Emas Theo Mandel

* Menempatkan pengguna sebagai pengendali
* Mengurangi beban memori pengguna
* Membuat antarmuka yang besifat konsisten

Perancangan Prosedural

Hasil Perancangan Prosedural

Rancangan detil tiap modul dan fungsi.

Notasi yang Digunakan

* Flowcharts
* Box Diagrams
* Decision Table
* Program Design Language

Kriteria Penilaian

Modularity

* Notasi mendukung pengembangan perangkat lunak.

Overall simplicity

* Mudah di pelajari, digunakan, dan ditulis.

Ease of editing

* Mudah untuk dimodifikasi ketika terjadi perubahan.

Machine readability

* Notasi dapat diinput langsung kedalam sistem.

Maintainability

* Mantenance konfigurasi melibatkan pemeliharaan desain procedural.

Structure enforcement

* Menggunakan pemrograman terstruktur.

Automatic processing

* Mengizinkan perancang untuk memverifikasi kebeneran dan kualitas perancangan.

Data representation

* Kemampuan untuk merepresentasikan data lokal dan global secara langsung.

Logic verification

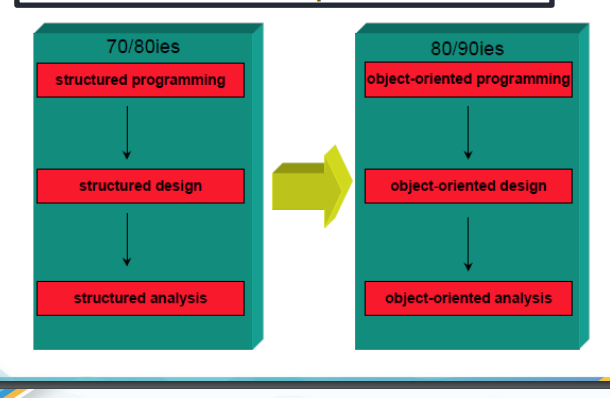
* Verifikasi logika otomatis meningkatkan kecukupan pengujian.

Easily converted to program source cod

* Membuat pembuatan kode lebih cepat.

4 - Analisis Berorientasi Objek

Evolusi Metode Berorientasi Objek



1. Metode beorientasi objek mulai berkembang ketika Grady Booch pada tahun 80-an mempublikasikan suatu paper bagaimana melakukan perancangan untuk bahasa ADA namun memberi judul paper tersebut Object Oriented Design.
2. Selanjutnya ide tersebut terus ia kembangkan sampai tahun 90 an.
3. Pada tahun 1991 Peter Coad dan Yourdon memperkenalkan metode berorientasi objek yang lebih sederhana dibandingkan Booch.
4. Metode ini menjadi cepat populer karena mendukung layanan-layanan yang terdapat pada C++.
5. Pada waktu itu C++ merupakan bahasa pemrograman berorientasi objek yang paling populer.
6. Pada tahun 1994 Ivar Jacobson memperkenalkan konsep use case dan object oriented software engineering.
7. Pada tahun 1994 itu juga yaitu bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, mempelopori usaha untuk penyatuan notasi pendekatan berorientasi objek.
8. Pada tahun 1995 dihasilkan draft pertama dari UML (versi 0.8).
9. Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG – http://www.omg.org).
10. Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003.
11. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999.
12. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

Analisis Berorientasi Obyek

Berfokus pada pendefinisian kelas-kelas dan cara bagaimana mereka saling bekerjasama satu dengan yang lainnya untuk memenuhi kebutuhan para pelanggan.

Pada Paradigma Analysis Design dan Diagram, UML merupakan tools yang digunakan untuk melakukan pemodelan berorientasi objek.

UML – Unified Modelling Language

Pengertian UML

* Digunakan untuk memodelkan semua proses dalam siklus hidup pengembangan dan seluruh teknologi implementasi yang berbeda.
* Bahasa standar untuk memvisualisasikan,menspesifiksi, konstruksi, dan mendokumentasikan artifak dari sistem perangkat lunak.
* Suatu alat komunikasi untuk team dan para stakeholders.

Sejarah UML

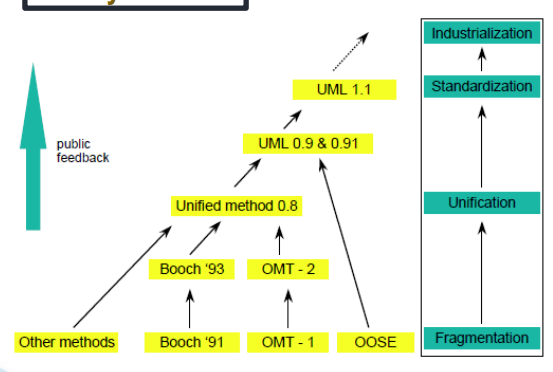
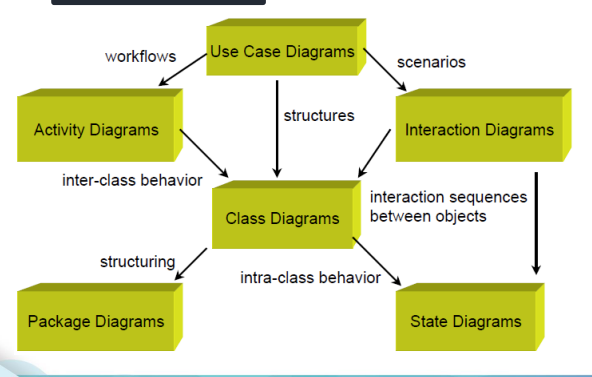
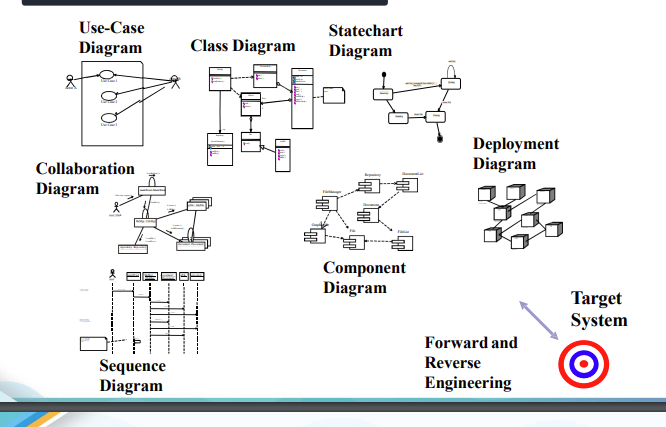


Diagram UML





UML Structure Diagrams (UML 2.0)

Pengertian UML Structure Diagrams

Mewakili data dan hubungan statis pada sistem informasi.

Contoh Structure Diagrams:

* Class Diagrams

Kosakata umum yang digunakan oleh analis dan pengguna. Mewakili sesuatu / benda (employee, paycheck, dll). Menunjukkan hubungan antar kelas.

* Object Diagrams

Mirip dengan Class Diagram. Gambaran tentang objek-objek dalam sistem. Hubungan antar objek

* Package Diagrams

Kelompok elemen-elemen UML digunakan untuk membentuk tingkat konstruksi yang lebih tinggi.

* Deployment Diagrams

Menunjukkan arsitektur fisik dan komponen perangkat lunak sistem. Contoh network nodes.

* Component Diagrams

Hubungan fisik di antara komponen perangkat lunak Example – Client / Server (mesin mana yang berjalan pada software yang mana).

* Composite Structure

Menggambarkan struktur internal dari kelas yang kompleks.

UML Behaviour Diagrams (UML 2.0)

Pengertian UML Behaviour Diagrams

Menggambarkan hubungan dinamis antara objek yang mewakili sistem informasi bisnis.

Contoh Behaviour Diagrams:

* Activity Diagrams

Model proses pada suatu sistem informasi. Contoh: Business workflows, business logic.

* Sequence Diagrams

Urutan berdasarkan waktu interaksi.

* Communication Diagrams

Komunikasi antara sekumpulan objek yang berkolaborasi dari suatu aktivitas.

* Interaction Diagrams

Menunjukkan interaksi antar objek.

* Timing Diagrams

Menunjukkan bagaimana suatu objek berubah dari waktu ke waktu.

* Behaviour State Machine

Memeriksa perilaku dari suatu kelas. Menunjukkan model keadaan-keadaan yang berbeda dan transisi keadaan dari suatu objek.

* Protocol State Machine

Untuk mengkespresikan protocol penggunaan atau siklus hidup beberapa classifier.

* Use Case Diagrams

Menunjukkan interaksi antara sistem dan lingkungan. Menangkap kebutuhan bisnis.

Tools Pembuatan UML

* Enterprise Architect
* Rational Rose
* Visual Paradigm
* Microsoft Visio
* Star UML
* Netbeans UML Plugin

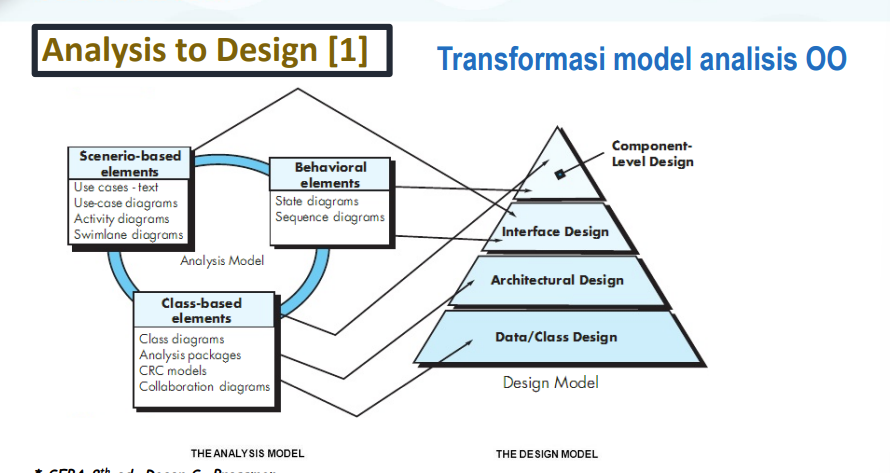
5 - Desain Berorientasi Objek

Perbedaan Class dan Objek

|  |  |
| --- | --- |
| Class | Object |
| konsep dan deskripsi dari sesuatu | Object: instance dari class, bentuk (contoh) nyata dari class |
| Class mendeklarasikan method yang dapat digunakan (dipanggil) oleh object | Object memiliki sifat independen dan dapat digunakan untuk memanggil method |
| Contoh:  mobil | Contoh:  Mobil berwarna merah, mobil pak Joko, mobilku |

Class seperti cetakan kue, dimana kue yang dihasilkan dari cetakan kue itu adalah object.

Analysis to Design



Software Design Engineering

Maksud desain yaitu mengumpulkan kebutuhan stakeholder, keperluan bisnis dan pertimbangan teknologi untuk memformulasikan suatu produk / sistem. Memodelkan aktivitas dan persiapan untuk tahap konstruksi (coding dan testing) tujuannya untuk memodelkan solusi yang siap diimplementasikan (membuat program).

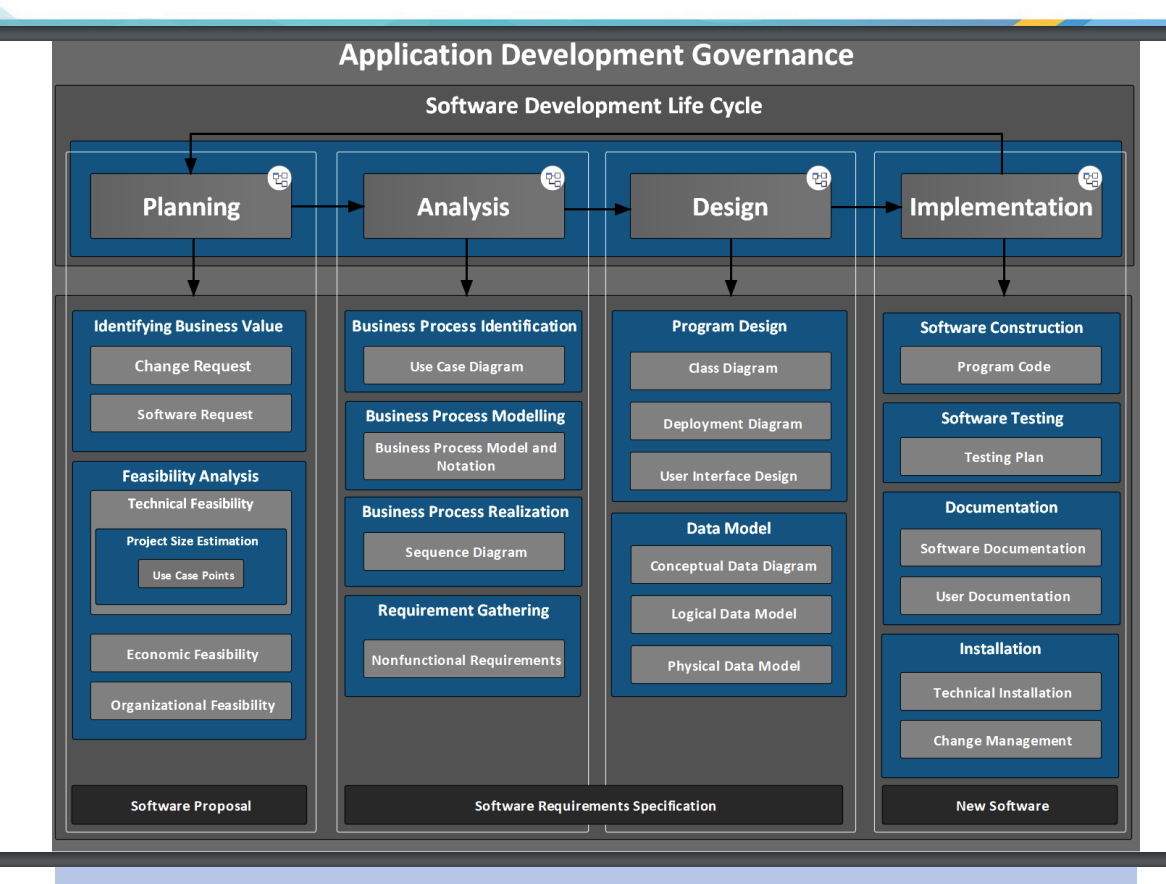
Proses Desain

Proses iteratif untuk menerjemahkan kebutuhan menjadi “blueprint” untuk membangun perangkat lunak.

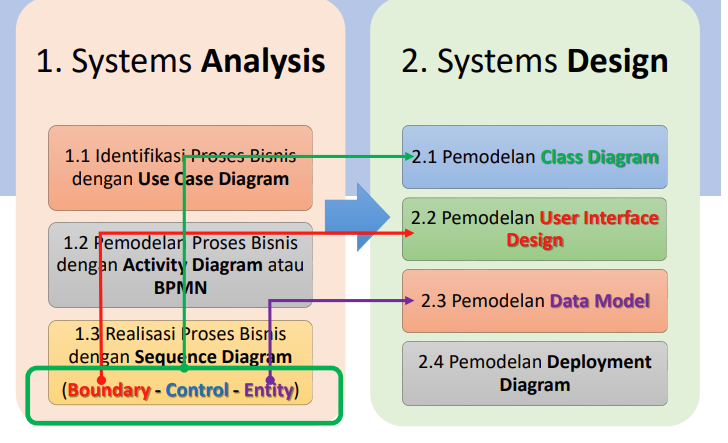
Karakteristik untuk mengevaluasi desain yang baik:

* Desain harus mengimplementasikan seluruh kebutuhan baik yang eksplisit dan implisit.
* Desain harus mudah dibaca dan dipahami.
* Desain harus menyediakan gambaran lengkap suatu perangkat lunak.

Software Development Lifecycle



UML Based Software Analysis and Design



Pemodelan Class Diagram

Class vs Package

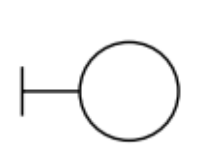
* Class

Penjelasan tentang satu set objek yang berbagi tanggung jawab yang sama, hubungan, operasi, atribut, dan semantik.

* Package

Mekanisme umum untuk mengatur elemen dalam kelompok-kelompok. Sebuah model elemen yang dapat berisi elemen model lainnya.

Boundary Class



* Digunakan untuk memodelkan interaksi antara sistem dan aktor.
* Sering mewakili windows, forms, sensors, terminals.
* Terkait dengan setidaknya satu aktor dan sebaliknya.

Control Class



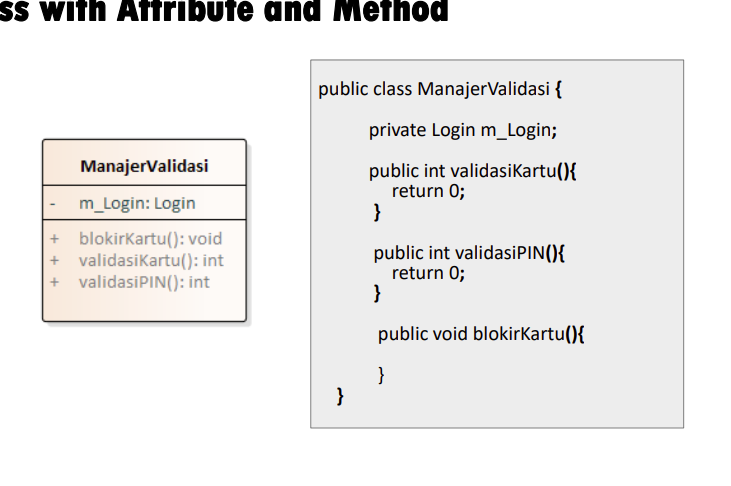
* Merepresentasikan koordinasi, urutan, transaksi dan pengendalian benda-benda lain.
* Sering merangkum use case yang terkait control.
* Satu use case memiliki satu control class.

Entity Class

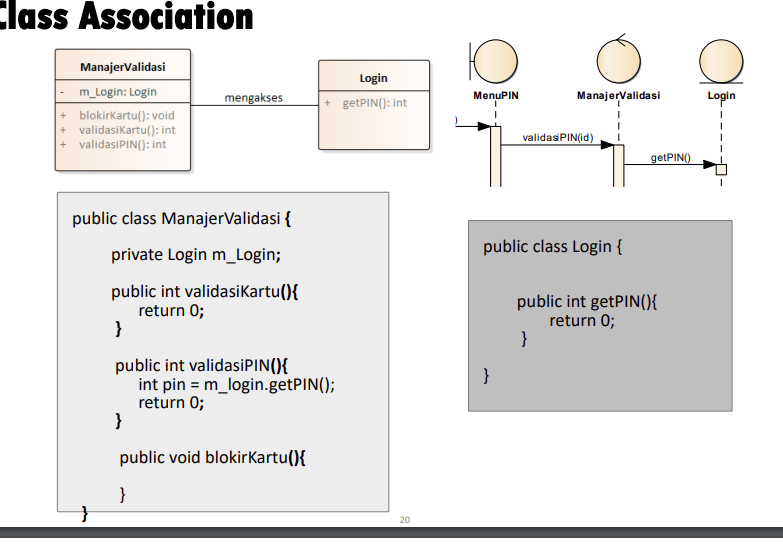


* Digunakan untuk memodelkan informasi.
* Biasanya berumur panjang / persisten.
* Biasanya berasal langsung dari badan usaha.
* Dapat dibagi oleh batas dan kontrol beberapa kelas.

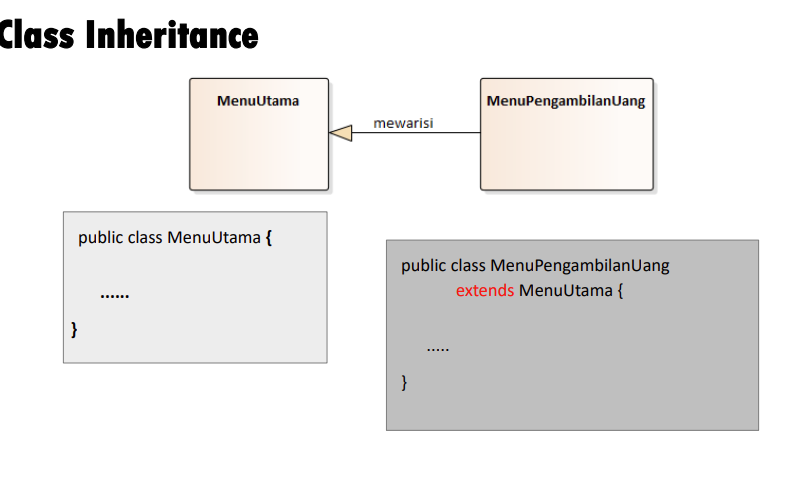
Class with Attribute and Method



Class Association



Class Inheritance



Pemodelan User Interface Design

Filosofi Interface Design

Desain antarmuka adalah seni.

Keseimbangan antara membuat antarmuka berguna dengan menyajikan banyak informasi.

Prinsip Interface Design

* Layout

Penggunaan area layar secara konsisten.

* Content Awareness

Pengguna tahu di mana mereka berada.

* Aesthetics

White space vs fungsionalitas.

* User Experience

Kemudahan penggunaan vs learning curve.

* Consistency

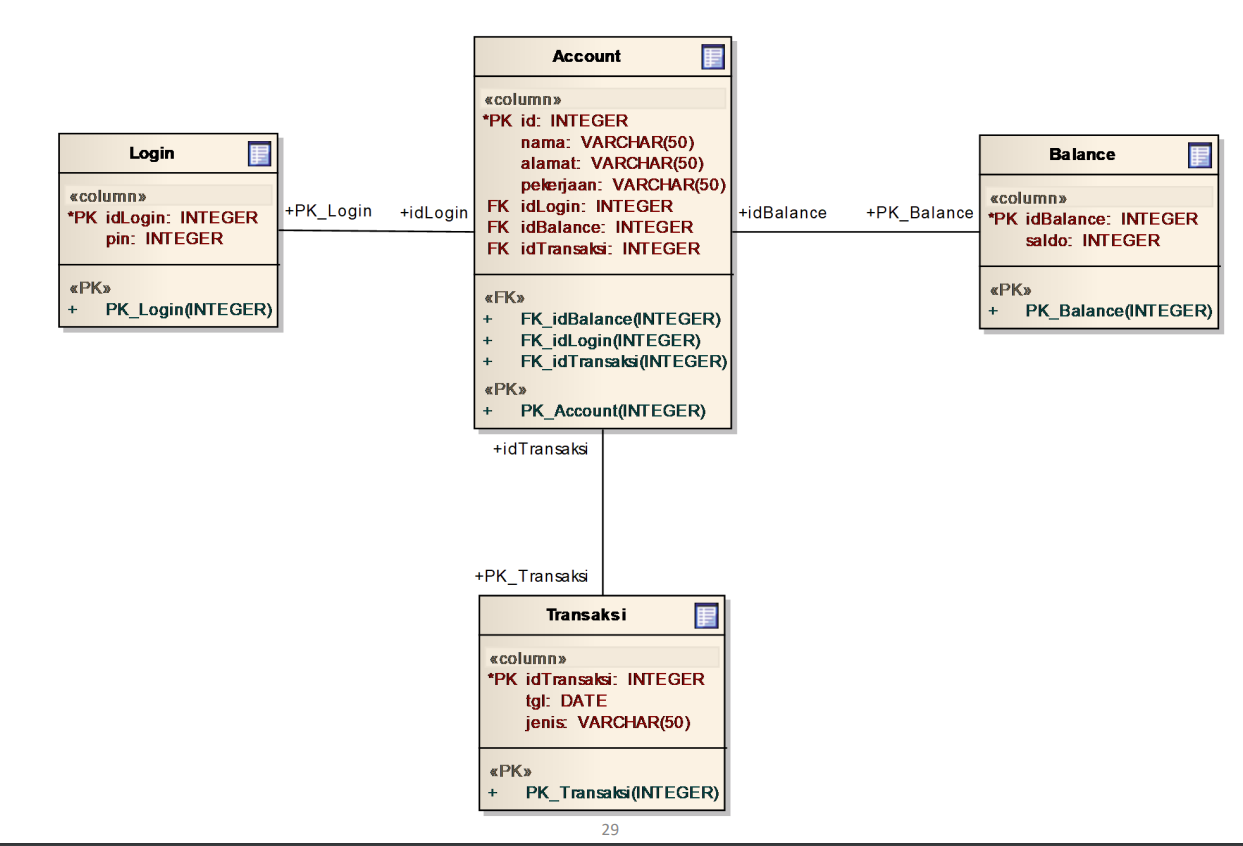
Pengguna dapat memprediksi apa yang akan terjadi untuk setiap tindakan.

* Minimal User Effort

Mudah digunakan, three click rule.

Pemodelan Data Model

Data Model Sistem ATM

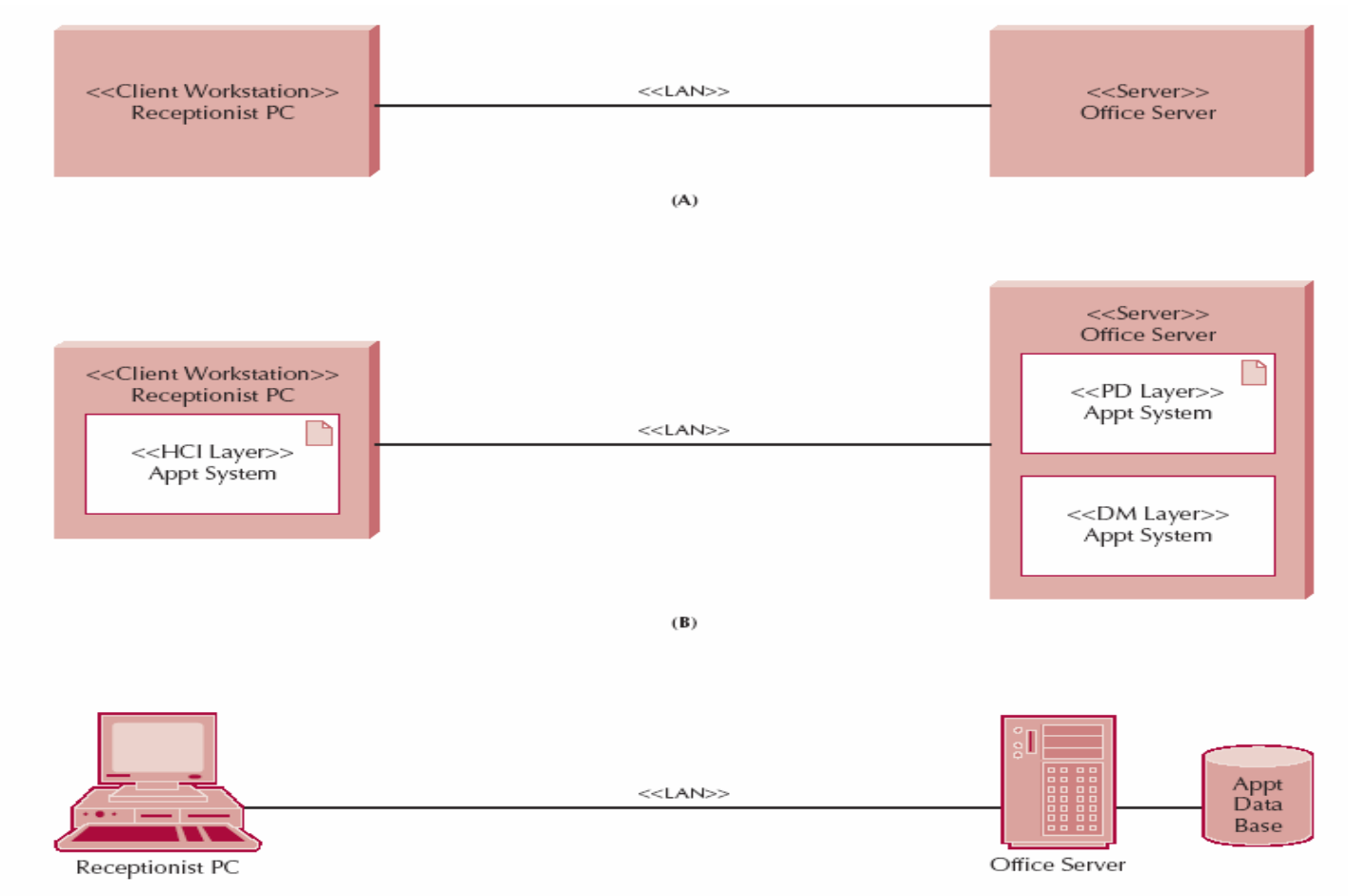


Pemodelan Deployment Program

Deployment Diagram

* Servers
* Mainframes, Minis, Micros
* Clients
* Input/Output HW yang digunakan oleh pengguna.
* Terminal, PC, HW tujuan khusus.
* Network
* HW dan SW untuk menghubungkan klien ke server.
* Nodes
* Setiap bagian dari perangkat keras dalam model.
* Sumber daya komputasi.
* Diberi label namanya.
* Stereotip untuk memberi label jenis simpul.
* Artifacts
* Bagian dari sistem informasi, seperti perangkat lunak atau tabel database.
* Node with Delpoyment Artifact
* Menunjukkan artefak yang ditempatkan pada physical node.
* Baik untuk menampilkan data distribusi atau perangkat lunak .
* Communication Paths
* Hubungan antara node jaringan.

Contoh Diagram



Deployment Diagram (3 Tier)

