Pertemuan 4

KONSEP PERANCANGAN

1. PENDAHULUAN

- Perancangan PL merupakan tempat dimana aturan kreativitas (kebutuhan *stakeholder*, kebutuhan bisnis, dan pertimbangan teknis) semuanya secara bersamaan disatukan untuk membentuk sebuah produk atau sistem/PL.
- Perancangan menciptakan representasi atau model PL.
- Model perancangan memberikan detail tentang arsitektur PL, struktur data, antarmuka, dan komponen yang diperlukan untuk mengimplementasikan sistem.
- Tujuan perancangan PL adalah untuk menghasilkan model atau representasi PL yang memperlihatkan kekuatan, komoditi, dan kenyamanan.

Model Perancangan

1. **Perancangan data/kelas**
   - Mengubah model kelas menjadi realisasi kelas perancangan dan struktur data yang diperlukan untuk mengimplementasikan PL.
   - Objek, hubungan dan konten data rinci yang digambarkan oleh atribut kelas dan notasi lainnya memberikan dasar untuk aktivitas perancangan data.
   - Bagian dari perancangan kelas dapat terjadi bersamaan dengan perancangan arsitektur PL.
   - Perancangan kelas yang lebih rinci terjadi karena setiap komponen PL dirancang.

Model Perancangan

2. Perancangan arsitektur

- Mendefinisikan hubungan antara elemen struktural utama dari PL, gaya dan pola arsitektur yang dapat digunakan untuk mencapai kebutuhan yang ditentukan untuk sistem, dan kendala yang mempengaruhi cara dimana arsitektur dapat diimplementasikan.
- Mewakili perancangan kerangka kerja arsitektur sistem berbasis komputer berasal dari model kebutuhan.

Model Perancangan

3. **Perancangan antarmuka**
   - Menggambarkan bagaimana PL berkomunikasi dengan sistem, dan dengan manusia yang menggunakannya.
   - Antarmuka menyiratkan aliran informasi (misal: data atau kontrol) dan jenis perilaku tertentu.
   - Perancangan antarmuka pada tingkat komponen mengubah elemen struktural dari arsitektur PL menjadi deskripsi prosedural komponen PL.
   - Informasi yang diperoleh dari model berbasis kelas dan model perilaku berfungsi sebagai dasar untuk perancangan komponen.

2. PROSES PERANCANGAN

- Perancangan PL adalah proses yang bersifat iteratif dimana spesifikasi kebutuhan PL diterjemahkan menjadi "*blue print*" untuk membangun PL.
- *Blue print* menggambarkan pandangan holistik PL. Yaitu, perancangan diwakili pada tingkat abstraksi yang tinggi pada tingkat yang dapat langsung ditelusuri ke tujuan sistem dan data yang lebih rinci, fungsional, dan kebutuhan perilaku.
- Saat iterasi perancangan PL berlangsung, penghalusan lebih lanjut akan menggerakkan abstraksi yang lebih rendah.

A. Atribut-Atribut Kualitas PL

Tiga karakteristik umum yang berfungsi sebagai panduan untuk evaluasi perancangan yang baik:

1. Perancangan PL harus menerapkan semua spesifikasi kebutuhan yang secara eksplisit ada dalam model kebutuhan, dan mengakomodasi semua spesifikasi kebutuhan implisit yang diinginkan oleh *stakeholder*.
2. Perancangan PL harus menghasilkan produk kerja yang mudah dibaca dan dimengerti bagi mereka yang membuat kode-kode program dan yang akan melakukan mengujian untuk kemudian mendukung PL.
3. Perancangan PL seharusnya menyediakan gambaran lengkap tentang PL, menangani permasalahan data, fungsional, dan perilaku dari perspektif implementasi.

Panduan Kualitas PL

Tujuannya untuk mengevaluasi kualitas representasi perancangan.

1. Rancangan PL harus menunjukkan arsitektur yang:
   - telah dibuat menggunakan gaya atau pola arsitektur yang dapat dikenali
   - tersusun atas komponen-komponen yang menunjukkan karakteristik perancangan yang baik
   - dapat diimplementasikan secara evolusioner, dan memfasilitasi implementasi dan pengujian.
2. Rancangan PL harus bersifat modular, artinya PL harus secara logis menjadi bagian dalam elemen-elemen atau subsistem.

Panduan Kualitas PL (Lanjutan)

3. Rancangan PL harus berisi representasi data, arsitektur, objek-objek, antarmuka, dan komponen yang berbeda.
4. Rancangan PL harus memuat struktur data yang sesuai untuk kelas yang akan diimplementasikan dan digambarkan dari pola-pola data yang dapat dikenali.
5. Rancangan PL harus mengarah pada komponen yang menunjukkan karakteristik fungsional yang bersifat independen.
6. Rancangan PL harus memuat antarmuka yang mengurangi kompleksitas hubungan antar komponen dan dengan lingkungan eksternal.

Panduan Kualitas PL (Lanjutan)

7. Rancangan PL harus diturunkan dari metode perulangan yang dikendalikan oleh informasi yang diperoleh selama analisis kebutuhan PL.
8. Sebuah perancangan harus direpresentasikan menggunakan notasi yang secara efektif mengkomunikasikan maknanya.

Atribut-Atribut Kualitas

1. **Fungsionalitas**
   Dinilai dengan mengevaluasi sejumlah fitur dan kemampuan program, fungsi-fungsi umum yang disampaikan, dan keamanan sistem secara keseluruhan.

2. **Peggunaan**
   Dinilai dengan mempertimbangkan faktor manusia, estetika, konsistensi, dan dokumentasi.

3. **Keandalan**

   Dievaluasi dengan mengukur frekuensi dan tingkat keparahan kegagalan, akurasi hasil keluaran, waktu antar kegagalan, *recovery*, dan prediktabilitas program.

Atribut-Atribut Kualitas (Lanjutan)

4. **Kinerja**
   Diukur menggunakan kecepatan pemrosesan, waktu tanggap, konsumsi sumber daya, *throughput*, dan efisiensi.

5. **Daya dukung**
   Menggabungkan kemampuan program untuk dikembangkan, kemampuan beradaptasi, dan kemampuan melayani kebutuhan pengguna, kemampuan uji, kompatibilitas, konfigurabilitas (kemampuan untuk mengatur dan mengontrol elemen-elemen konfigurasi PL).

3. KONSEP-KONSEP PERANCANGAN

Konsep perancangan pada dasarnya akan menyediakan kebutuhan:

- Kriteria yang digunakan untuk membagi PL menjadi komponen yang bersifat mandiri
- Rincian fungsi/struktur data yang dapat dipisahkan dari representasi konseptual
- Kriteria yang digunakan untuk mendefinisikan kualitas teknis perancangan PL

A. Abstraksi

- Pada tingkat abstraksi tertinggi, penyelesaian masalah dinyatakan dalam istilah luas menggunakan bahasa pada lingkungan permasalahan.
- Pada tingkat abstraksi lebih rendah, deskripsi yang lebih rinci dari penyelesaian masalah harus diberikan.
  - Terminologi berorientasi masalah digabungkan dengan terminologi berorientasi implementasi dengan tujuan untuk lebih dapat menyelesaikan permasalahan
- Pada tingkat abstraksi yang paling rendah, penyelesaian masalah dapat langsung diimplementasikan menggunakan bahasa pemrograman yang dipilih.

B. Arsitektur

- Arsitektur sistem/PL merupakan struktur keseluruhan PL dan cara bagaimana struktur tersebut memberikan integritas konseptual untuk suatu sistem/PL.
- Arsitektur sistem/PL juga merupakan struktur/organisasi dari komponen program (modul) yang menjelaskan bagaimana komponen-komponen tersebut berinteraksi, dan struktur data yang digunakan oleh komponen.

Properti sebagai bagian dari perancangan arsitektur:

- **Properti Struktural**

  Mendefinisikan komponen sistem (modul, objek, filter) dan menjelaskan bagaimana komponen-komponen tersebut dikemas dan berinteraksi satu sama lain.

- **Properti Fungsional Tambahan**

  Deskripsi perancangan arsitektural seharusnya bisa menyelesaikan permasalahan tentang bagaimana arsitektur perancangan mencapai kebutuhan akan kinerja, kapasitas, keandalan, keamanan, kemampuan beradaptasi, dan karakteristik lainnya.

- **Keluarga sistem yang berhubungan**

  Perancangan arsitektural seharusnya dibuat melalui pola perulangan/iterasi.

C. Pola (*Pattern*)

- Pola adalah suatu wawasan yang di dalamnya memuat esensi dari solusi yang terbukti untuk permasalahan perancangan PL dalam konteks tertentu [Brad Appleton].
- Tujuan dari setiap pola perancangan adalah untuk memberikan deskripsi yang memungkinkan perancang untuk menentukan:

  1. Apakah pola berlaku untuk pekerjaan saat ini
  2. Apakah pola dapat digunakan kembali
  3. Apakah pola dapat berfungsi sebagai panduan untuk mengembangkan pola yang serupa, tetapi berbeda secara fungsional atau struktural

D. Pemisahan Perhatian

- Pemisahan perhatian adalah konsep perancangan yang menunjukkan bahwa masalah yang kompleks dapat diselesaikan jika permasalahan itu dibagi menjadi bagian-bagian yang lebih kecil yang lebih mudah diselesaikan dan/atau dioptimalkan secara independen.
- Pemisahan perhatian dapat diimplementasikan menggunakan konsep perancangan PL yang saling berhubungan seperti: modularitas, aspek-aspek, kemandirian fungsional, dan penghalusan.

E. Modularitas

- Modularitas adalah atribut tunggal dari PL yang memungkinkan suatu program untuk dapat dikelola secara cerdas.
- PL monolitik adalah program besar yang terdiri dari satu modul.
- PL monolitik tidak dapat dengan mudah dipahami oleh rekayasawan PL. Jumlah lintasan kendali, rentang referensi, jumlah variabel, dan kompleksitas keseluruhan hampir tidak mungkin dimengerti.

Modularitas (Lanjutan)

- Lebih banyak modul berarti ukuran modul semakin lebih kecil.
- Bertambahnya jumlah modul, maka upaya (biaya) yang terkait dengan pengintegrasian modul juga bertambah.
- Tujuan Modularitas:
  a. Pengembangan PL dapat lebih mudah direncanakan
  b. Versi-versi PL dapat didefinisikan
  c. Perubahan-perubahan dapat dengan mudah diakomodasikan
  d. *Testing* dan *debugging* dapat dilakukan secara lebih efisien
  e. Mudah dalam pemeliharaan PL

F. Penyembunyian Informasi

- Maksudnya bahwa Modul PL harus dispesifikasikan dan dirancang sedemikian rupa sehingga informasi (algoritma dan data) yang terdapat dalam modul tidak dapat diakses oleh modul lain yang tidak memerlukan informasi tersebut.
- Penyembunyian informasi (*Information hiding*) menyiratkan bahwa modularitas yang efektif dapat dicapai dengan mendefinisikan sejumlah modul independen yang berkomunikasi satu sama lain dalam hal informasi yang diperlukan untuk mencapai fungsi PL.
- Manfaatnya ketika modifikasi tertentu perlu dilakukan selama pengujian PL dan selama pemeliharaan PL.

G. Independensi Fungsional

- Konsep independensi fungsional adalah hasil langsung dari pemisahan masalah, modularitas, dan konsep abstraksi dan penyembunyian informasi.
- Independensi fungsional dicapai dengan cara mengembangkan modul yang memiliki fungsi tunggal (*single-minded*) dan memiliki interaksi yang bersifat tertutup dengan modul lain.
- Modul independen lebih mudah dikembangkan karena fungsi-fungsi di dalamnya dapat dilokalisasi dan antarmuka disederhanakan.
- Modul independen lebih mudah dipelihara dan diuji

Independensi Fungsional (Lanjutan)

Modul independensi mempunyai kriteria kualitatif:

1. **Kohesivitas**
   - Adalah indikasi dari kekuatan fungsional suatu modul.
   - Umumnya melakukan pekerjaan tunggal dan hanya memerlukan sedikit interaksi dengan komponen lainnya

2. **Kebergantungan (*Coupling*)**
   - Adalah indikasi dari kemandirian antar modul.
   - *Coupling* tergantung pada kompleksitas antarmuka yang menghubungkan modul-modul yang ada dalam program, yaitu titik masuk suatu modul, dan data apa yang melewati antarmuka modul.

H. Penghalusan

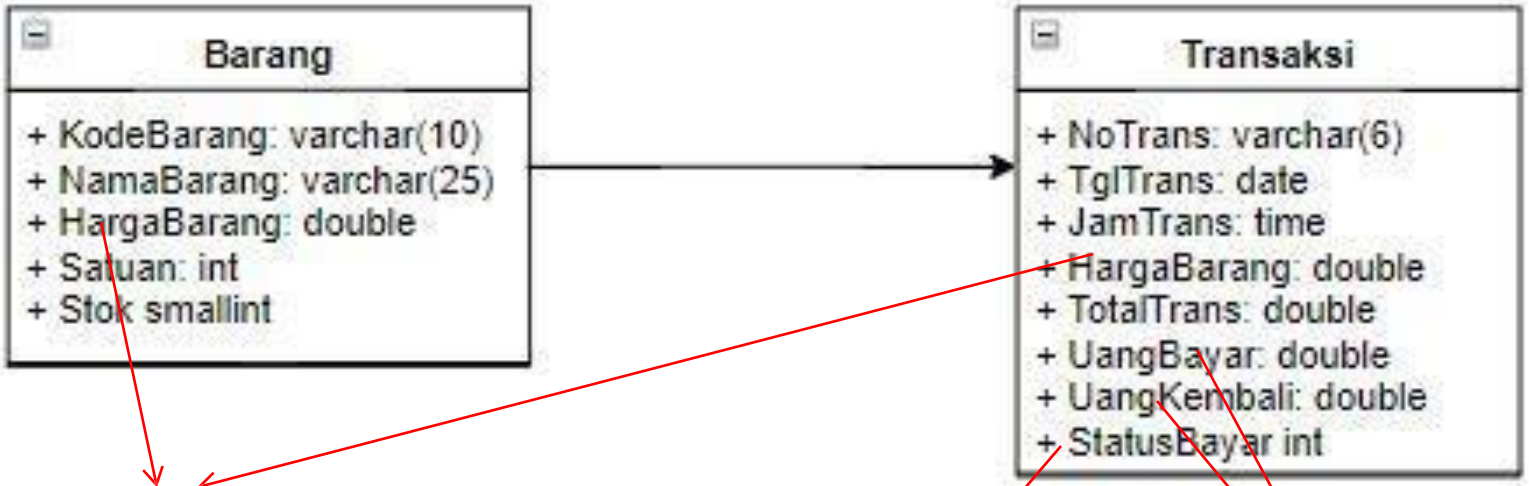
- Penghalusan merupakan proses elaborasi yang dimulai dengan pernyataan fungsi (atau deskripsi informasi) yang didefinisikan pada tingkat abstraksi yang tinggi.
- Penghalusan langkah demi langkah menggunakan strategi perancangan *top-down.*
- Hirarki PL dikembangkan dengan melakukan dekomposisi pernyataan fungsi yang bersifat makro (abstraksi prosedural) secara bertahap hingga pernyataan dalam bahasa pemrograman dapat tercapai.
- Penghalusan membantu untuk mendapatkan rincian pada tingkat rendah saat perancangan berlangsung.

I. Refaktorisasi

- Refaktorisasi adalah teknik reorganisasi perancangan PL yang bertujuan untuk menyederhanakan perancangan (atau kode) dari komponen tanpa harus mengubah fungsi atau perilakunya.

- Refaktorisasi PL, diperiksa kembali untuk:
  a. Menemukan redundansi
  b. Menemukan elemen perancangan yang tidak digunakan
  c. Menemukan algoritma yang tidak efisien (tidak perlu)
  d. Menemukan struktur data yang konstruksinya buruk
  e. Menemukan kegagalan perancangan lainnya yang dapat diperbaiki untuk menghasilkan perancangan yang lebih baik.

Contoh:

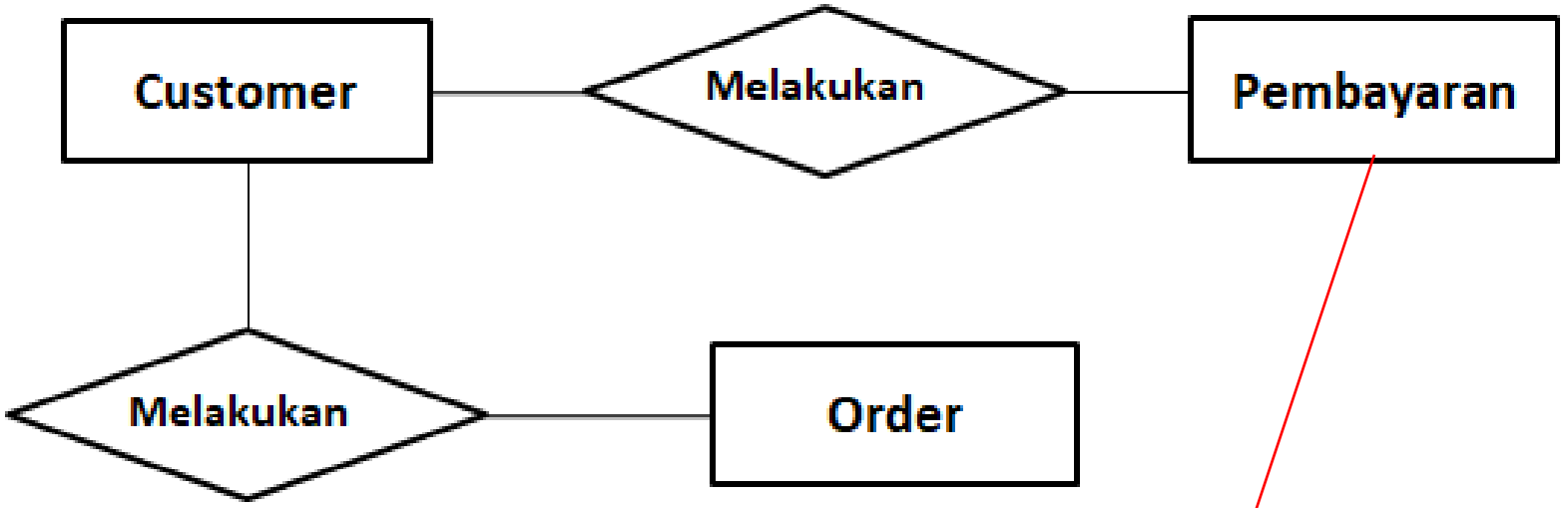


Pendefinisian data yang menyebabkan redundansi data pada tabel Transaksi. Harga barang dituliskan pada kedua tabel, bisa saja dengan kode yang sama tetapi harga berbeda.

Elemen data yang tidak berfungsi di dalam sistem

Elemen data StatusBayar yang tidak perlu digunakan, menyebabkan ambigu

Contoh:



Konstruksi database yang tidak baik.
Customer dapat melakukan pembayaran padahal belum melakukan Transaksi (Order)

