Tim Pengajar IF2250

IF2250 – Rekayasa Perangkat Lunak

Analisa Kebutuhan P/L Pendekatan Terstruktur

SEMESTER II TAHUN AJARAN 2023/2024

KNOWLEDGE & SOFTWARE ENGINEERING

Kontraktor/
(System Analyst)

Membuat
Kebutuhan sistem

Kebutuhan User dan Kebutuhan Sistem

User memberikan masukan lebih rinci

User

1. Requirements Gathering/Elicitation

2. Requirements Analysis/Classification

Requirements Specification (draft)

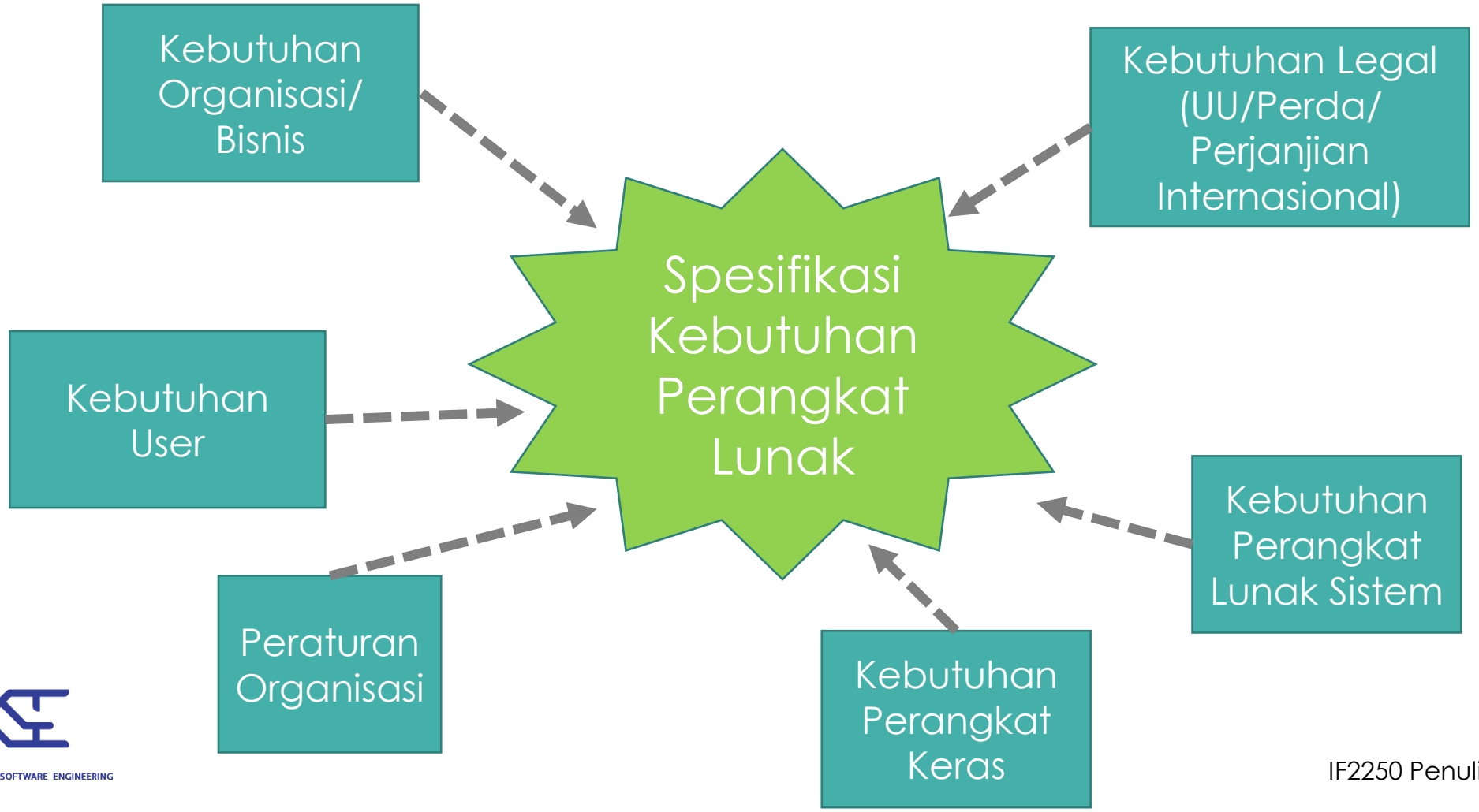
3. Requirements Validation/Negotiation

5. Requirements Management

4. Requirements Specification (final)

KNOWLEDGE & SOFTWARE ENGINEERING

Spesifikasi Kebutuhan PL dibuat berdasarkan berbagai Kebutuhan Sistem



KNOWLEDGE & SOFTWARE ENGINEERING

Analisis Kebutuhan

- Melakukan pengkajian ulang (**review**) terhadap semua dokumen yang relevan
- Mengembangkan konsep awal
- Konsultansi dengan pengguna
  - Mendapatkan kejelasan
  - Negosiasi jika diperlukan, misalnya karena
    - Ada kebutuhan yang mungkin di luar anggaran
    - Butuh waktu yang lebih lama karena mungkin alasan teknis/non teknis
    - Ada kebutuhan di luar batasan yang pernah di bicarakan sebelumnya.
- Konsolidasi semua data masukan ke sistem
- Konsolidasi semua hasil keluaran dari sistem

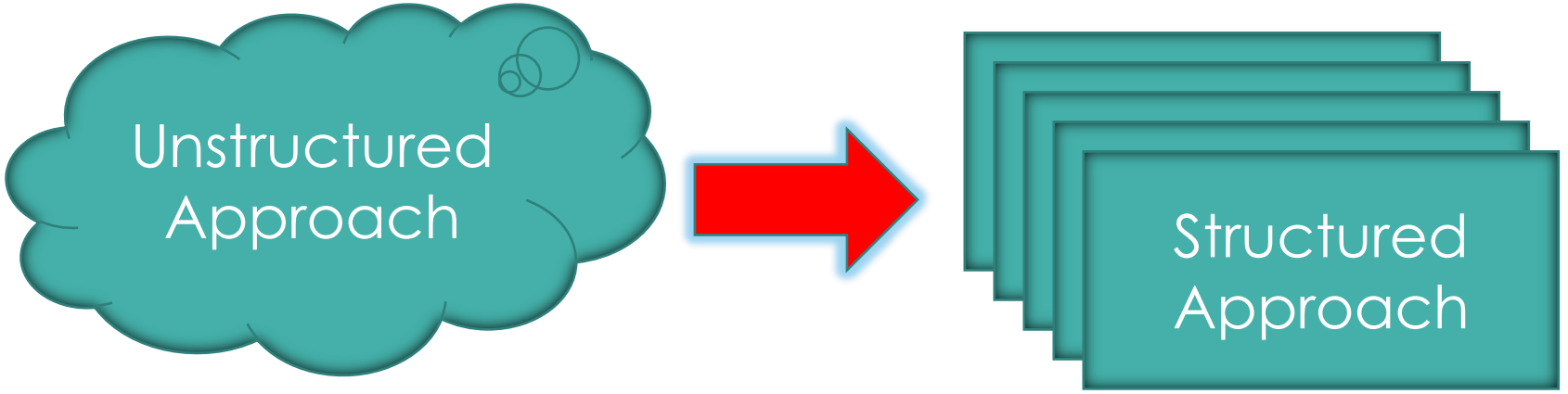
Pemodelan Kebutuhan

- Spesifikasi kebutuhan perlu diperjelas dengan suatu model atau diagram
- Pemodelan hingga saat ini menggunakan 2 pendekatan
  - Model dengan pendekatan Terstruktur (**Structured Approach**) – Tradisional/Konvensional
    - Diagram konteks – DFD
    - Diagram ER
  - Model dengan pendekatan Obyek (**Object Orientation**) – lebih baru, diagram UML (Unified Modeling Language)
    - Diagram Use case
    - Diagram Kelas/Objek
    - Diagram Sekuens, dll

Interaksi dengan Sistem

- Pada tahap awal pengembangan perangkat lunak kita harus mendefinisikan **siapa** atau **apa** saja yang terkait dengan sistem/perangkat lunak(software) yang akan dibuat.
- Pengembangan model diperlukan untuk memudahkan analisa dan juga komunikasi
  - Diagram konteks dapat digunakan untuk menggambarkan siapa saja yang berinteraksi dengan sistem
  - Use-case digunakan untuk menggambarkan keterlibatan ‘aktor’ dengan sistem yang akan dikembangkan.

Pendekatan Terstruktur



- Program relatif **skala kecil**
  - Persoalan masih sederhana
  - Jumlah individu terlibat sedikit
  - Kompleksitas persoalan rendah

- Program relatif skala **menengah ke atas**
  - Persoalan melibatkan banyak elemen terkait
  - Jumlah individu makin banyak
  - Kompleksitas persoalan tinggi
  - Butuh waktu yang tepat
  - Butuh usaha besar

KNOWLEDGE & SOFTWARE ENGINEERING

Dekomposisi Fungsional

- **Dekomposisi fungsional** adalah teknik untuk mendekomposisi fungsional dari perangkat lunak yang akan dibangun
- Dekomposisi ini bergantung pada fungsi atau aksi yang akan dilakukan oleh PL.
- Caranya:
  - Tulis dalam satu kalimat apa yang akan dilakukan oleh PL
  - Jika PL melakukan beberapa fungsi, tulis setiap fungsi dalam satu baris
- Contoh:
  - Ingin membuat program robot untuk membuat secangkir kopi
  - Dekomposisi
    - Robot dapat mengambil cangkir
    - Robot dapat menyeduh kopi
    - Robot dapat 'mengantar' kopi kepada pemesannya

Pemodelan Analisa Kebutuhan dengan Pendekatan Terstruktur

- *Data Flow Diagram*
- *Entity Relationship Diagram*
- *State Transition Diagram*

Analisis Terstruktur Data Flow Diagram

KNOWLEDGE & SOFTWARE ENGINEERING

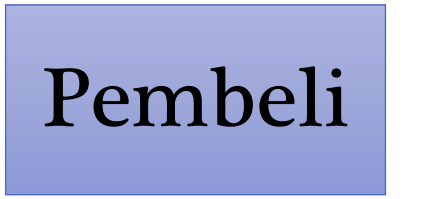
Analisis Aliran Data (Data flow analysis)

- Metode ini dapat/biasa digunakan sebagai dasar untuk merancang arsitektur P/L skala besar
- Caranya
  - Identifikasi **aliran data** yang akan terjadi pada PL
  - Identifikasi **transformasi** yang terkait dengan aliran data tsb
- Dengan metode ini, maka akan didapat
  - **Proses-proses** dalam PL
  - **Hubungan antar proses**
    - Dilihat dari aliran data dari suatu proses ke proses lain

Hubungan aktivitas lain

- Hasil proses analisis kebutuhan ini akan terkait dengan:
  - Perancangan Sistem Awal (*Preliminary Design*)
  - Perancangan Sistem
  - Manajemen Konfigurasi
  - Kualitas
  - Validasi

Notasi Penulisan DFD

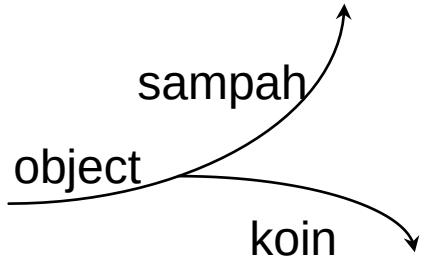


Terminator/
Entitas eksternal/
Bagian Input/output

Process

Data Store

Aliran Data



KNOWLEDGE & SOFTWARE ENGINEERING

Pengembangan DFD

- Dimulai dengan pengembangan **Diagram Konteks**
  - **Diagram Konteks** digunakan untuk menjelaskan **keterhubungan sistem** yang dikembangkan dengan **sistem lain** atau **aktor luar** atau entitas eksternal
    - Contohnya Aktor pada Vending Machine: Para pengguna VM, Pemilik VM
    - Aktor ini bisa 'manusia' atau 'sistem'
      - Contohnya: sistem PL untuk mesin ATM dapat saja berhubungan dengan Bank A, Bank B, dan Bank C. Jadi Aktor/Entitas Eksternal nya adalah Bank A, B dan C
  - Pada level berikutnya dibuatlah **DFD Level 0**, **Level 1**, dan seterusnya
  - **Kamus Data** (*Data Dictionary*) diperlukan untuk menjelaskan data atau komposisi dari suatu data

Diagram Konteks

- Konteks diagram menjelaskan fungsi sistem dan mendefinisikan **jangkauan dari sistem**
- Hanya ada **satu lingkaran** proses
- Minimal ada **satu input** dan **satu output**
- Minimal ada **satu entitas eksternal/terminator**
- Semua terminator terhubung minimal satu aliran input atau output
- Tidak ada pertukaran data antara terminator

Diagram Konteks (1)

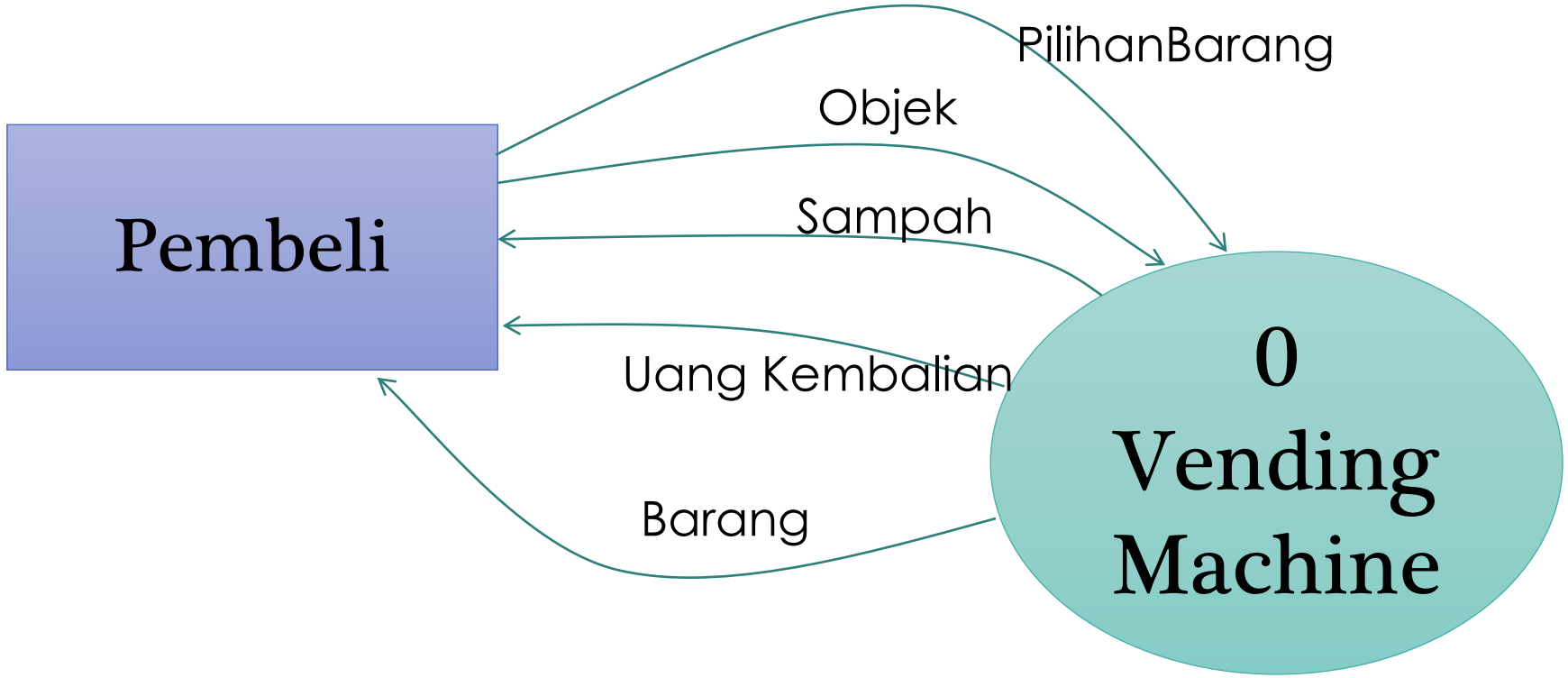


Diagram Konteks (2)

- Diagram konteks menunjukkan **pertukaran data** antara **sistem** dengan **lingkungannya**.
- Terminator ditunjukkan pada diagram konteks karena **terminator bukan bagian** dari sistem atau software yang akan dikembangkan
- **Data store** bukan bagian dari diagram konteks karena dianggap sebagai **bagian dari sistem**
- Diagram konteks hanya terdiri dari
  - **Satu proses** (sistem /software nya)
  - **Beberapa terminator/entitas eksternal**
  - **Aliran data**

Diagram Konteks (3)

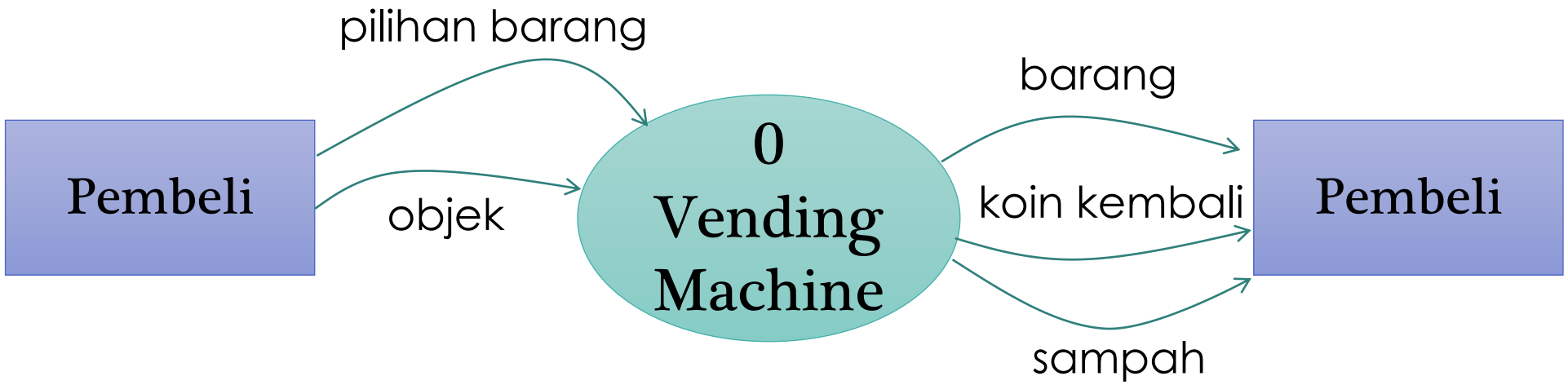


Diagram Konteks (4)

- Konteks diagram dibentuk secara **iteratif**
  - Terutama untuk pembentukan batasan sistem (system boundary)
- Untuk sistem yang kompleks, kadang kita harus menggambar dulu beberapa DFD sebelum batasannya (boundary) dapat kita kembangkan
- Batasan ini juga dapat berubah sesuai dengan kebutuhan yang baru muncul.

Dekomposisi Fungsional untuk DFD

- Kebutuhan fungsional didekomposisi, dengan mengikuti aturan:
  - Fungsi-fungsi yang **terkait disatukan** dalam suatu kelompok
  - Fungsi-fungsi yang **tidak terkait dipisahkan**
  - **Setiap fungsi** hanya deskripsikan **satu kali** saja.

Panduan DFD (1)

- Setiap lingkaran menggambarkan satu proses
- Setiap **proses** akan **didekomposisi** menjadi proses lain dalam satu DFD atau menjadi **algoritma**
- Gunakan aturan **7 ± 2**
  - Sebaiknya 5 sampai 9 proses dalam satu gambar diagram
  - Keseimbangan antara kurang dari 4 yang artinya kurangnya informasi atau terlalu banyak bila lebih dari 9 proses dalam satu diagram
- Umumnya dekomposisi mencapai 4 level
- Tiap diagram DFD adalah dekomposisi dari lingkaran orangtuanya.

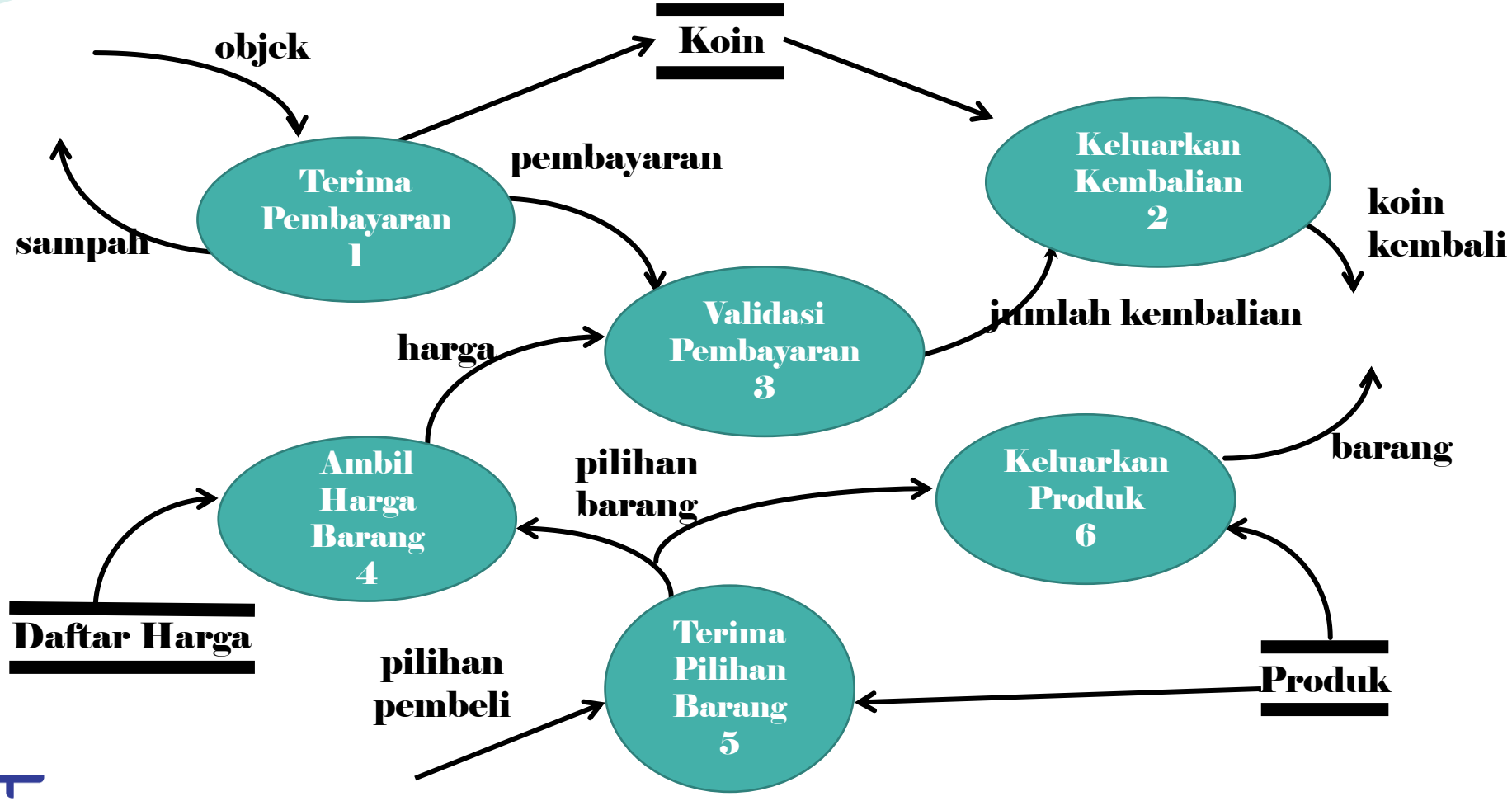
Panduan DFD (2)

- Proses-proses ini adalah **elemen** yang **aktif** dalam model
- Proses melakukan **transformasi**
  - Jika dan hanya jika semua **informasi keluaran** tersedia
  - Semua **transformasi** informasi bersifat **langsung**
- Keluaran dari suatu proses sebaiknya adalah suatu **hasil fungsi** dari inputnya
- Aliran data memungkinkan transitnya data dalam sistem
- Data Store adalah tempat penyimpanan informasi

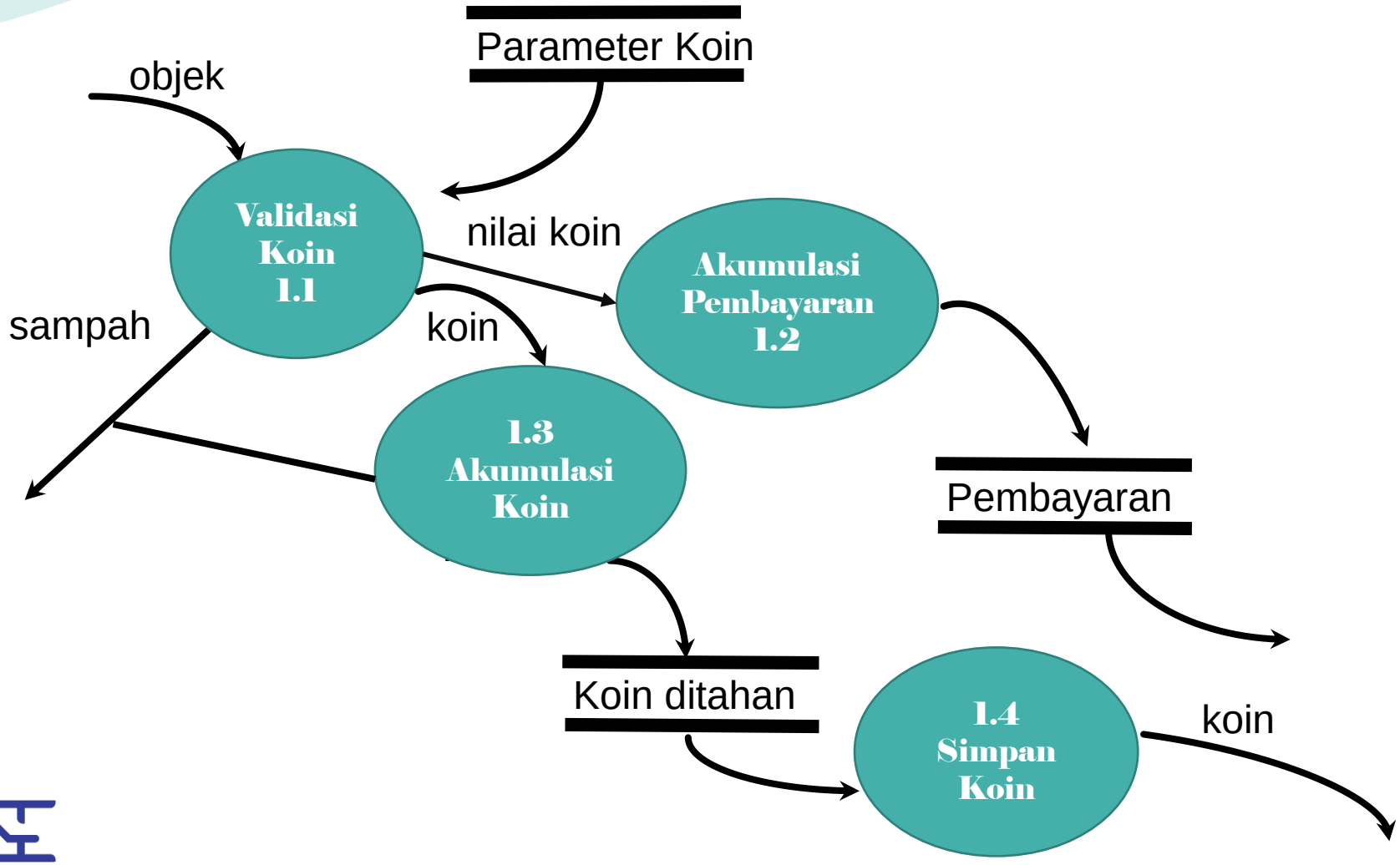
Panduan DFD (3)

- Balancing
  - Input dan output dari setiap proses hasil dekomposisi suatu proses harus sesuai dengan induknya
- Leveling
  - Proses dekomposisi disebut sebagai leveling
- Ketika suatu proses **tidak bisa** lagi **dipecah**, maka akan menjadi **primitif fungsional** dan akan diperjelas dengan **PSPEC**
  - PSPEC ditulis hanya untuk primitif fungsional

DFD Level 0


objek
Koin
Terima Pembayaran 1
pembayaran
sampah
Keluarkan Kembalian 2
koin kembali
Validasi Pembayaran 3
jumlah kembalian
harga
Ambil Harga Barang 4
pilihan barang
Keluarkan Produk 6
barang
Daftar Harga
Terima Pilihan Barang 5
pilihan pembeli
Produk

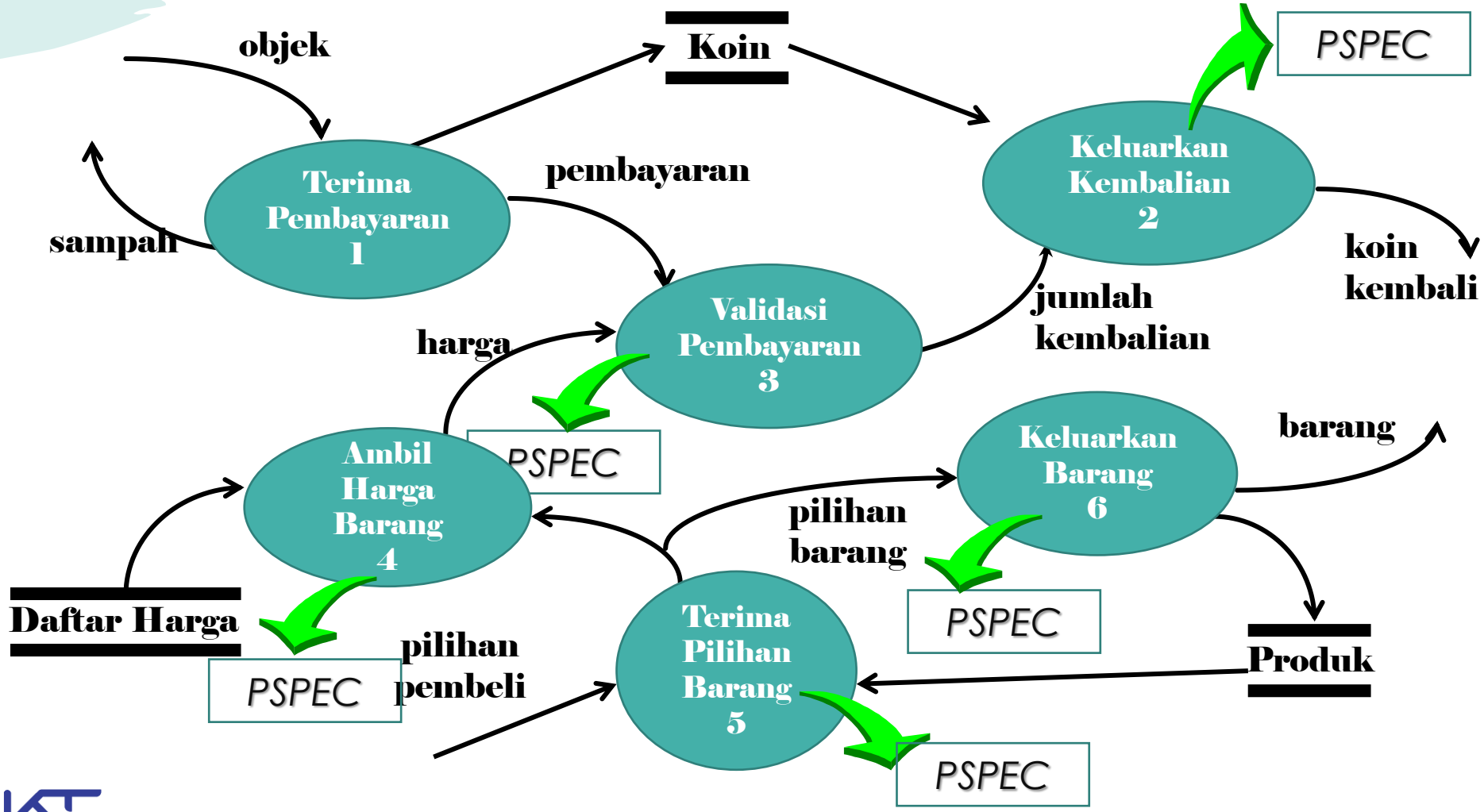
DFD Level 1: 1. Terima Pembayaran



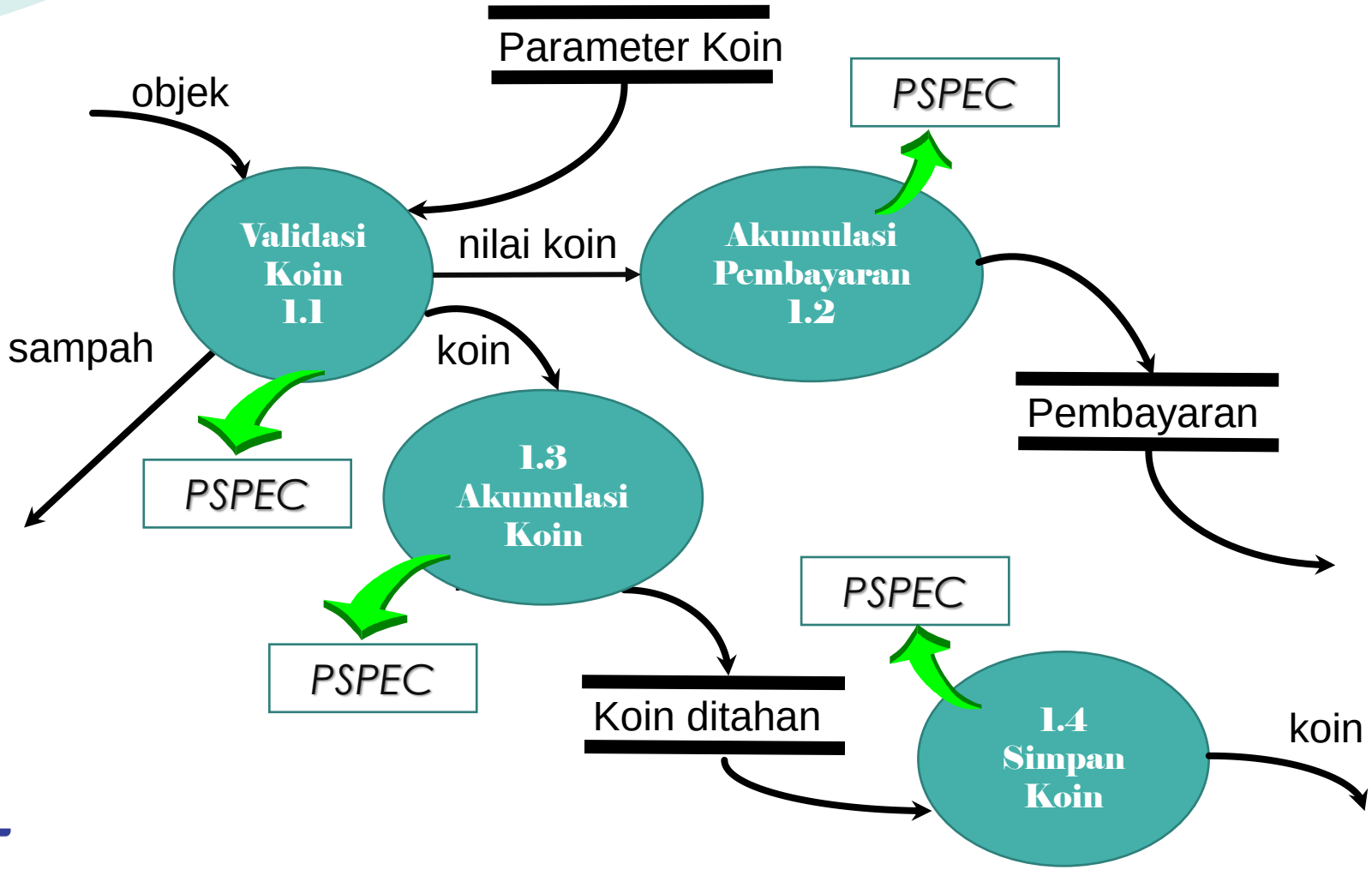
Spesifikasi Proses (PSPEC – Process Specification)

- PSPEC adalah **level abstraksi** yang paling **rendah** (di DFD)
- PSPEC sepanjang kira-kira ½ halaman
- Menunjukkan hubungan antara input proses dan aliran output
- Dapat menggunakan berbagai bentuk spesifikasi
  - Gambar
  - Persamaan matematika
  - Bahasa sehari-hari

Spesifikasi Proses (PSPEC – Process Specification)



DFD Level 1: 1. Terima Pembayaran



PSPEC 3: Validasi Pembayaran

Inputs:

pembayaran : data in
harga : data in

Outputs:

jumlah kembali : data out

Body:

```
If (pembayaran >= harga)
        jumlah kembalian= payment – harga
else
        jumlah kembalian = 0
```

PSPEC 2: Keluarkan Kembalian

Inputs:

koin : data in

jumlah kembali : data in

Outputs:

koin kembali : data out

Body:

koin kembali adalah jumlah koin yang diambil dari sebanyak ‘jumlah kembali’

PSPEC lainnya (1)

PSPEC 4: Ambil Harga Barang

Keluarkan harga barang berdasarkan pilihan barang yang diambil dari tabel daftar harga

PSPEC 5: Terima Pilihan Barang

Pilihan barang diambil berdasarkan pilihan pembeli jika produk yang dibeli ada dari daftar tabel 'produk'

PSPEC 6: Keluarkan Barang

Keluarkan Barang berdasarkan pilihan barang, update jumlah barang dari tabel produk

PSPEC lainnya (2)

PSPEC 1.1: Validasi Koin

Periksa Objek sesuaikan dengan data dari Parameter Koin.

Jika benar maka terima objek sebagai koin, jika tidak maka keluarkan objek sebagai sampah

PSPEC 1.2: Akumulasi Pembayaran

Tambahkan nilai koin ke data pembayaran.

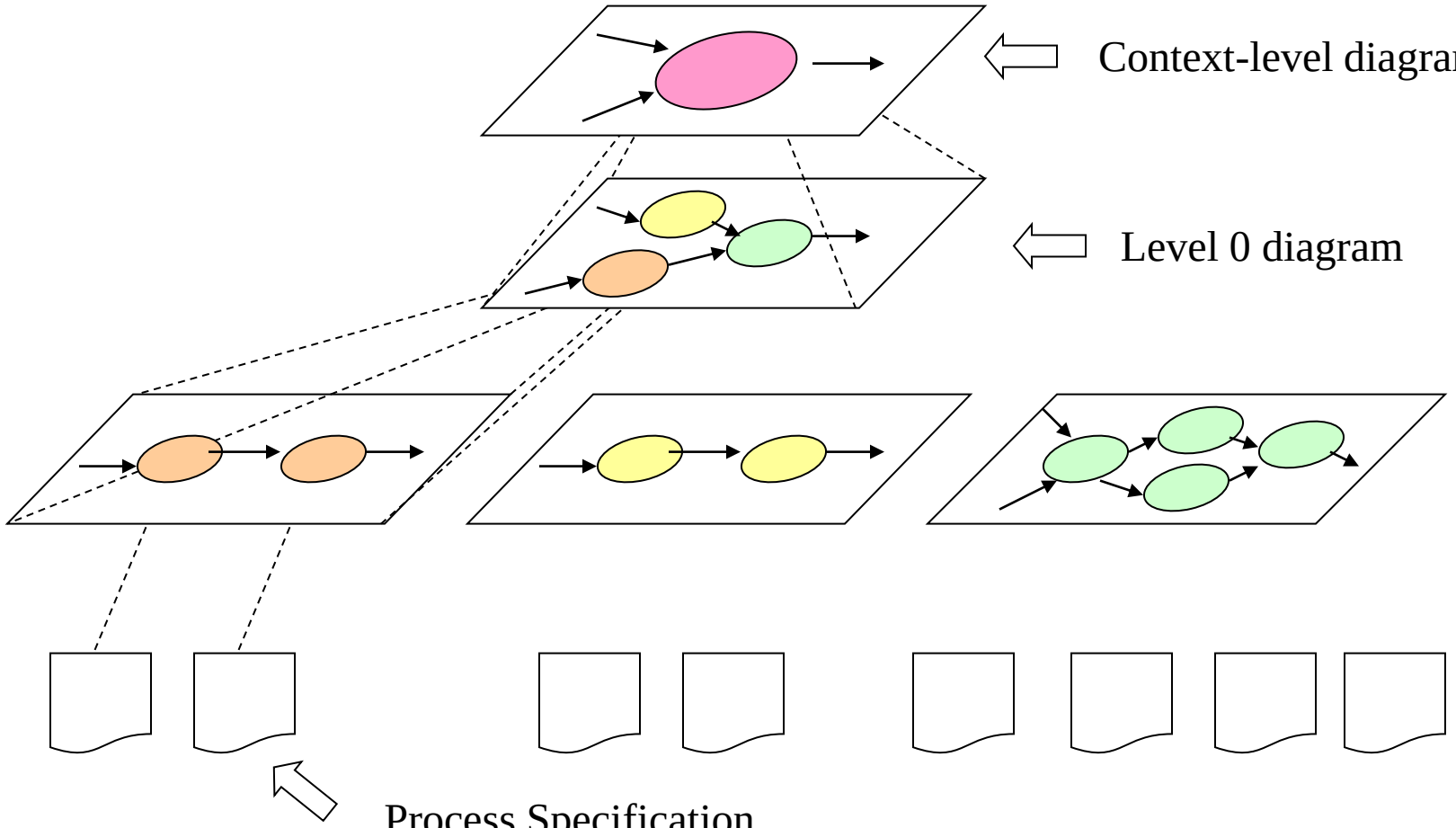
PSPEC 1.3: Akumulasi Koin

Bila jumlah koin melebihi kapasitas, maka koin dianggap sampah

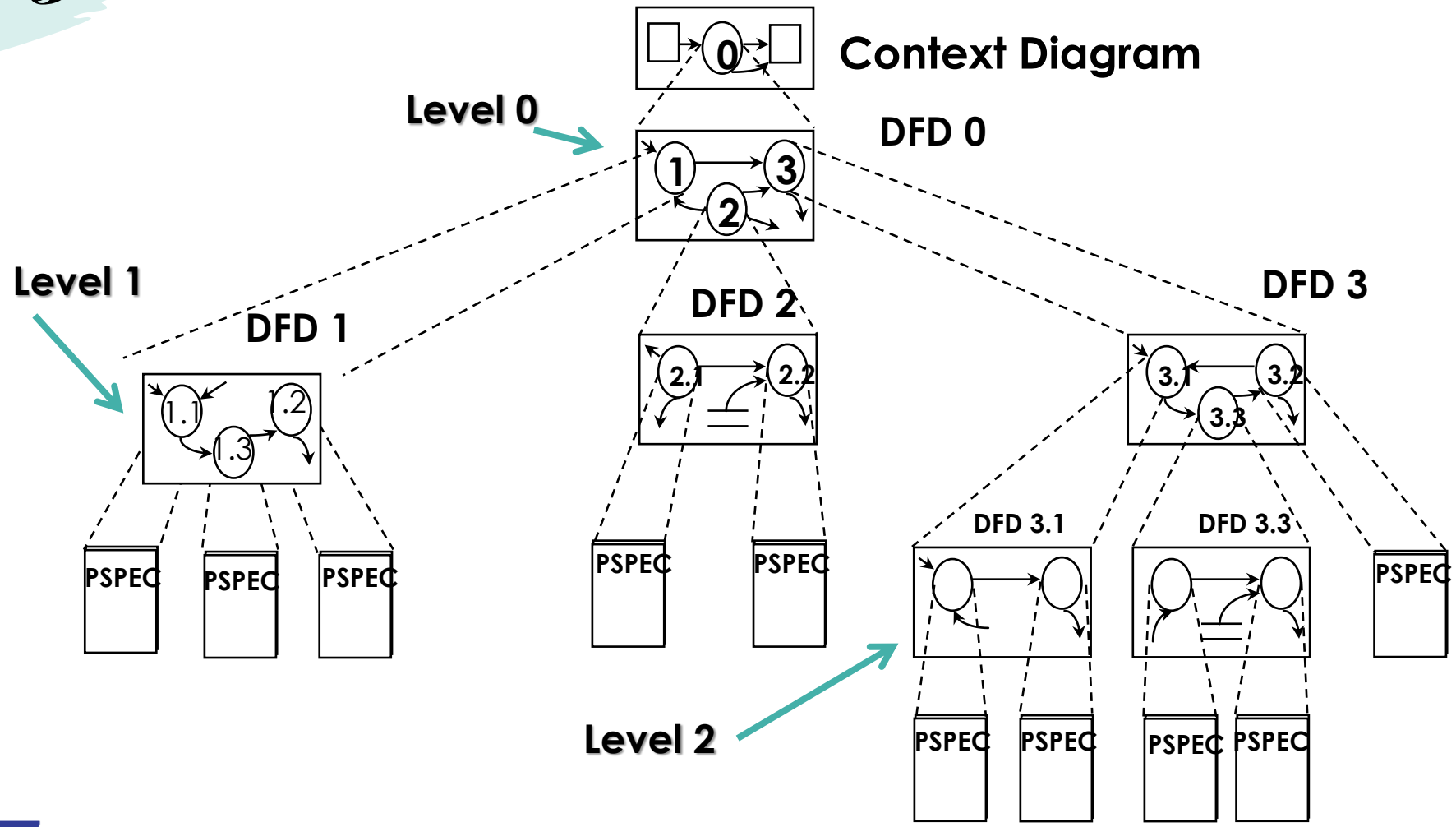
PSPEC 1.4: Simpan Koin

Pindahkan koin yang di tahan ke penampungan koin

Diagram Layering and Process Refinement



Rangkuman


Context Diagram
Level 0
DFD 0
Level 1
DFD 1
DFD 2
DFD 3
Level 2
DFD 3.1
DFD 3.3
PSPEC

Kamus Data (Data Dictionary)

- Berisi sekumpulan nama data dengan **definisi**nya
- Setiap **aliran data** harus **ada kamus**nya
- Nama-nama kelompok harus **dipecah** hingga bentuk yang paling **elementer**
- **Elemen primitif** memiliki atribut seperti: **unit**, **range**, **akurasi**
- **Kamus data** ini akhirnya akan menjadi bagian dari **basisdata** yang digunakan untuk mendukung pemodelan.

Aturan Penulisan

=	terdiri dari
+	digabungkan dengan
{ }	pengulangan
[..|..]	terdiri dari
( )	optional item
" "	literal
* *	memberikan insight tambahan tentang arti

Kamus Data (silakan dikembangkan lagi)

Nama	Deskripsi
Objek	= [koin | sampah]
Barang	= [soda | permen | chips]
koin	= [100an | 200an | 500an | 1000an]
100an	= *uang koin 100 rupiah*

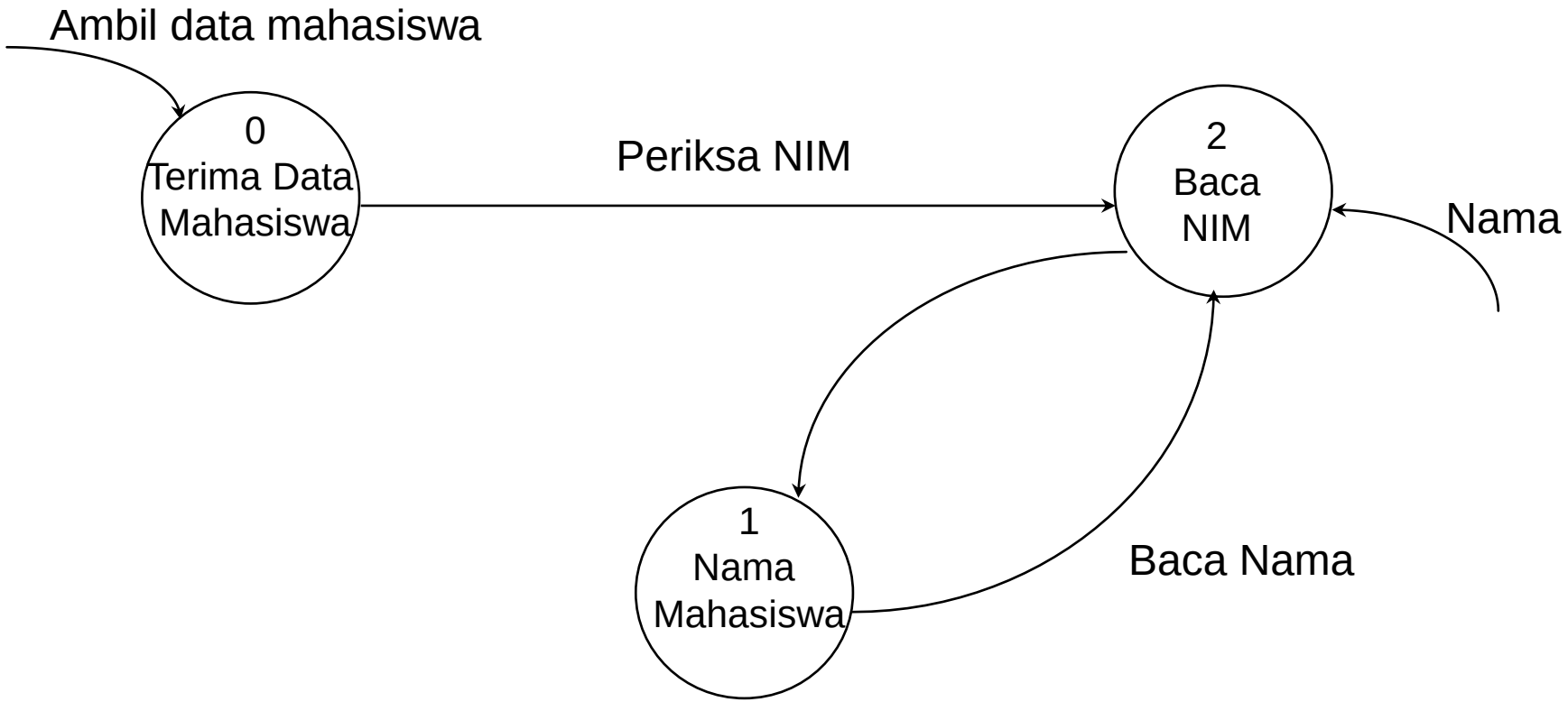
Aturan Balancing Kebutuhan

- Setiap DFD harus seimbang (balance) dengan induknya
- Setiap PSPEC harus seimbang dengan proses primitif fungsi yang terkait
- Setiap **aliran data**, dan **data store** harus terdefinisi, dan harus **terdekomposisi** menjadi **elemen primitif**
  - Elemen primitif ini akan didefinisikan di kamus data

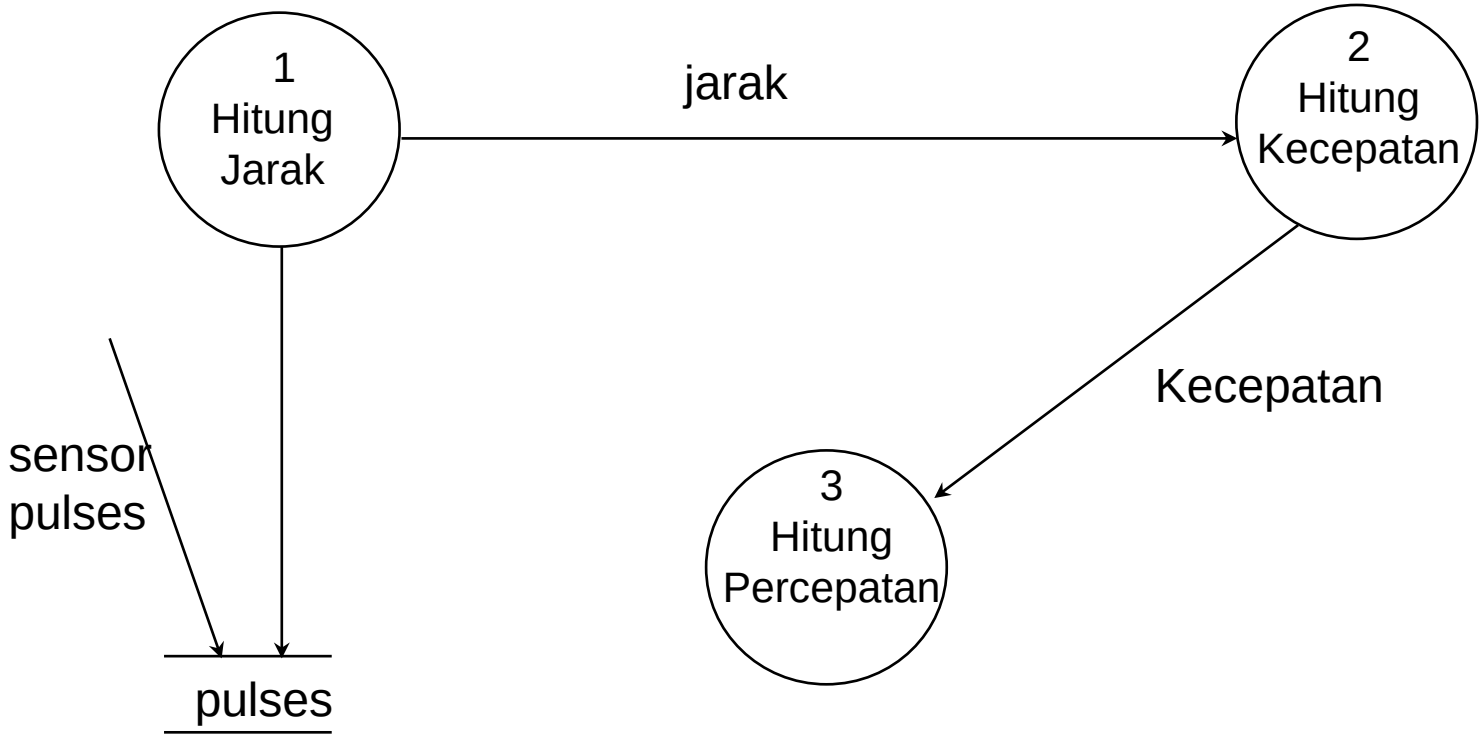
Evaluasi DFD

- Apakah semua **level** memang sudah yang **diperlukan**?
- Apakah **nama-nama** sudah **jelas**?
- Apakah setiap **bagian** sudah punya **nama** dan dapat dimengerti **makna**nya?
- Apakah ada aliran yang hilang?
- Apakah semua **proses** sudah sesuai dengan **kebutuhan** ?
- Apakah fungsi yang tidak terhubung sudah dipisahkan dan fungsi yang terhubung dikelompokkan bersama?

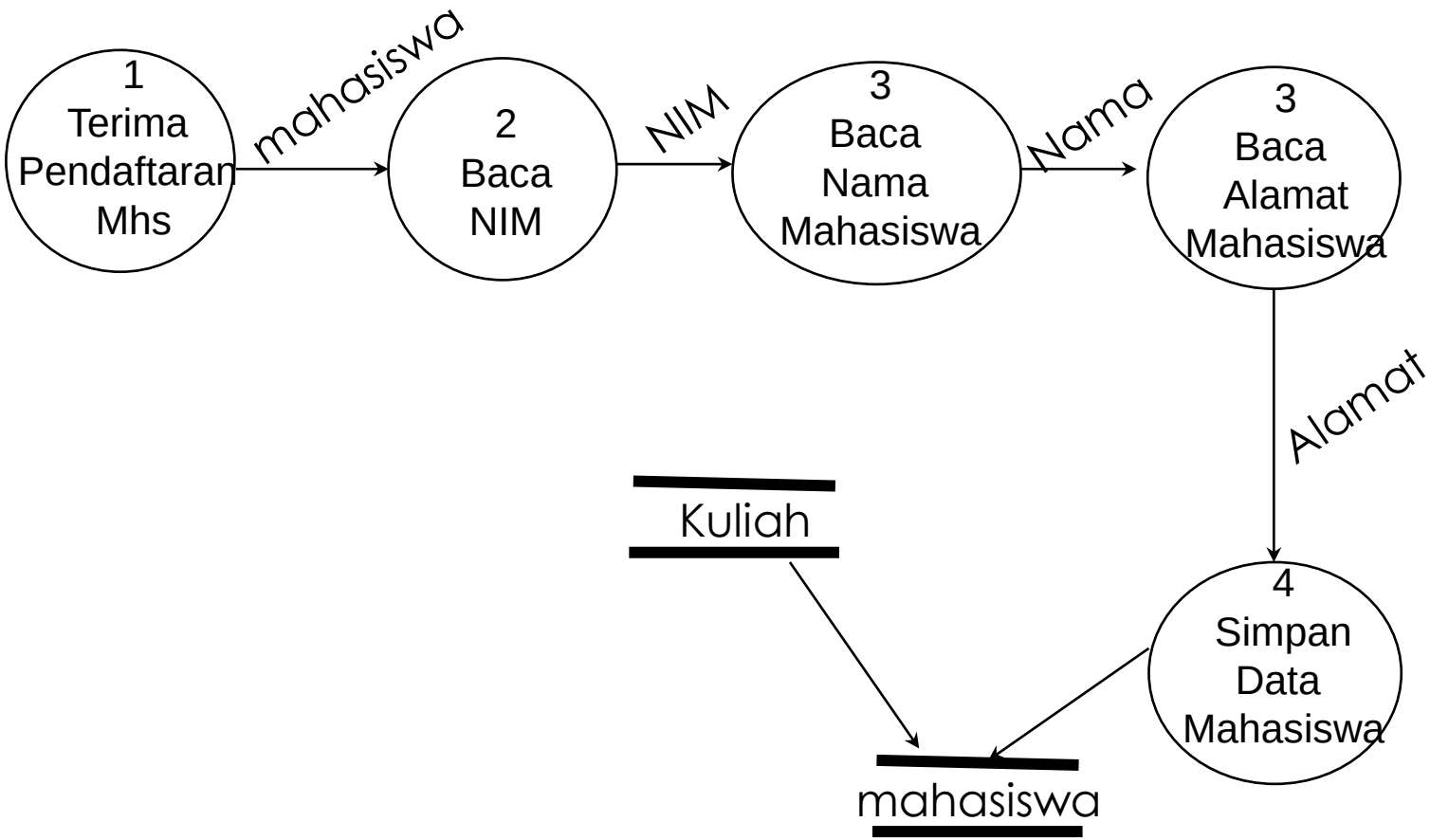
Latihan: Cari Salah (1)



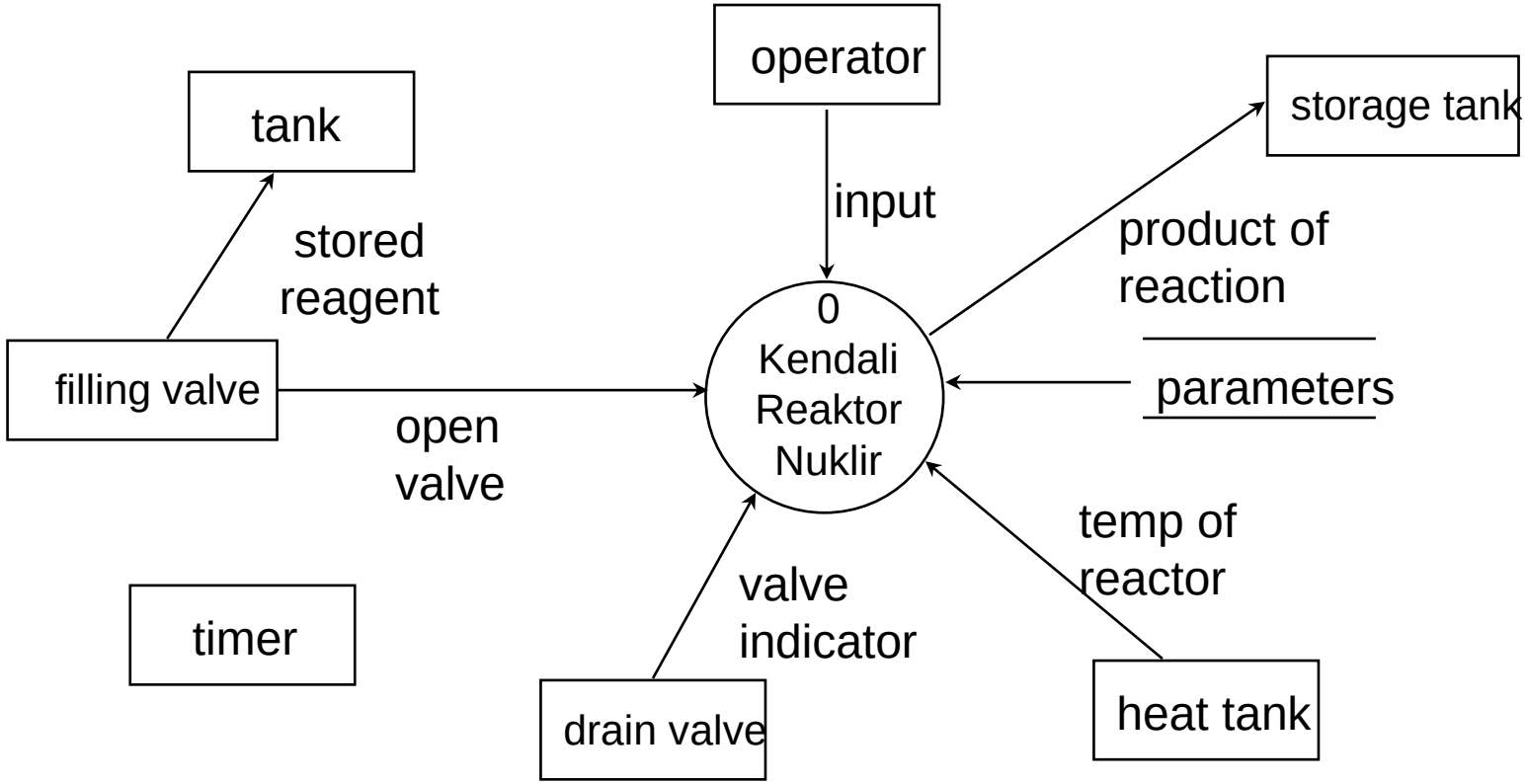
Latihan: Cari Salah (2)



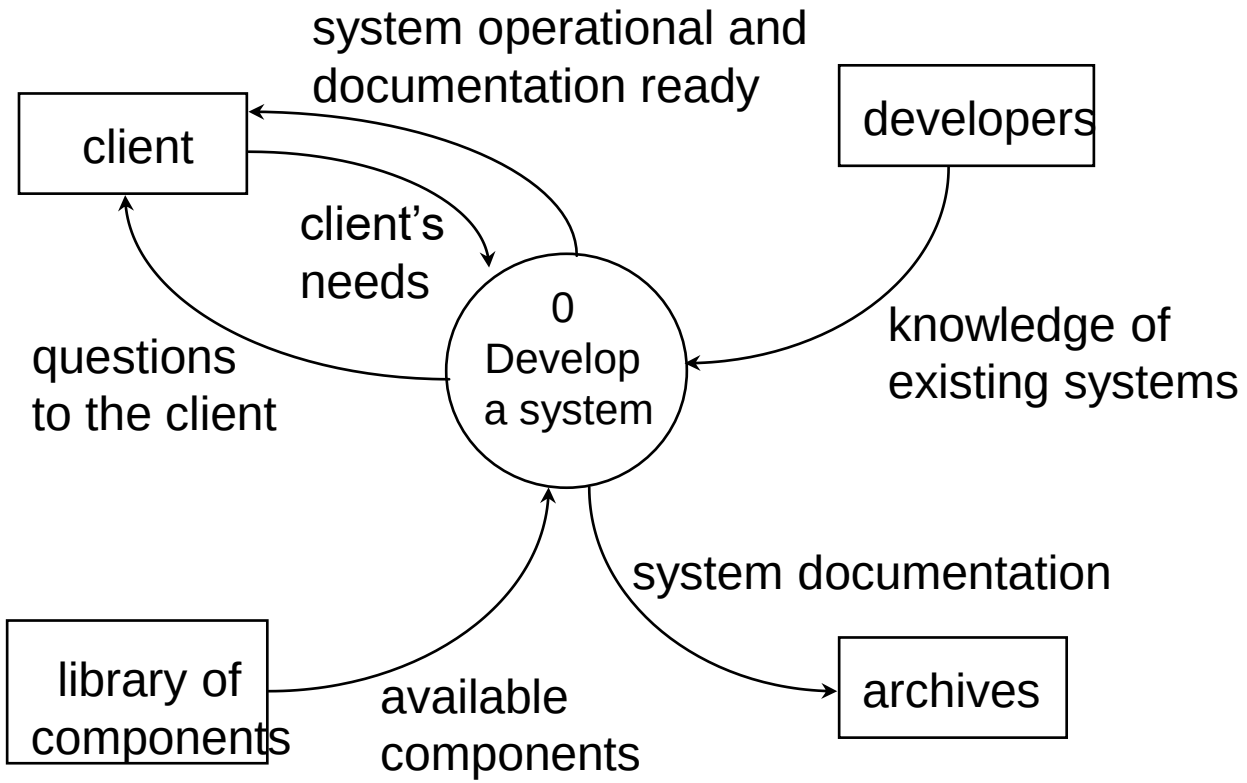
Latihan: Cari Salah (3)



Latihan: Cari Salah (4)

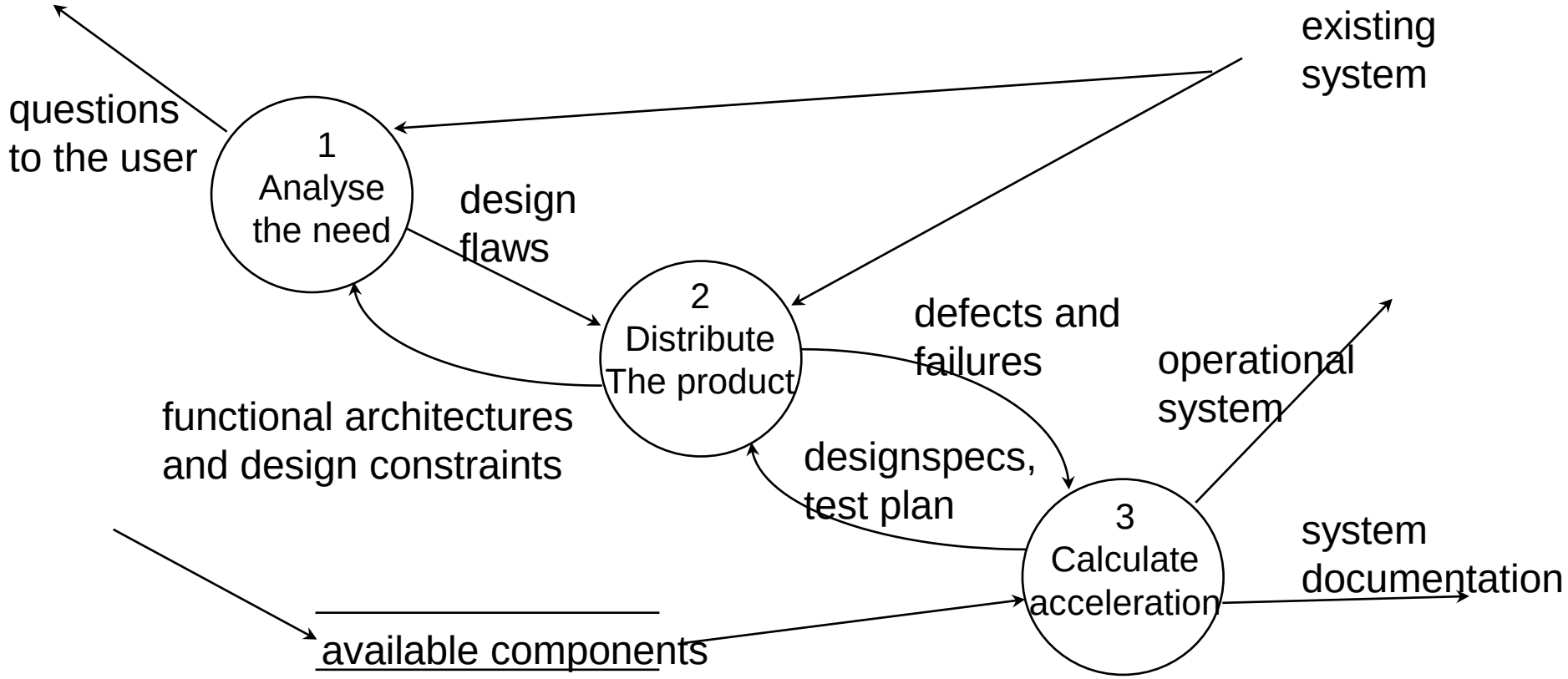


Latihan: Cari Salah (5)



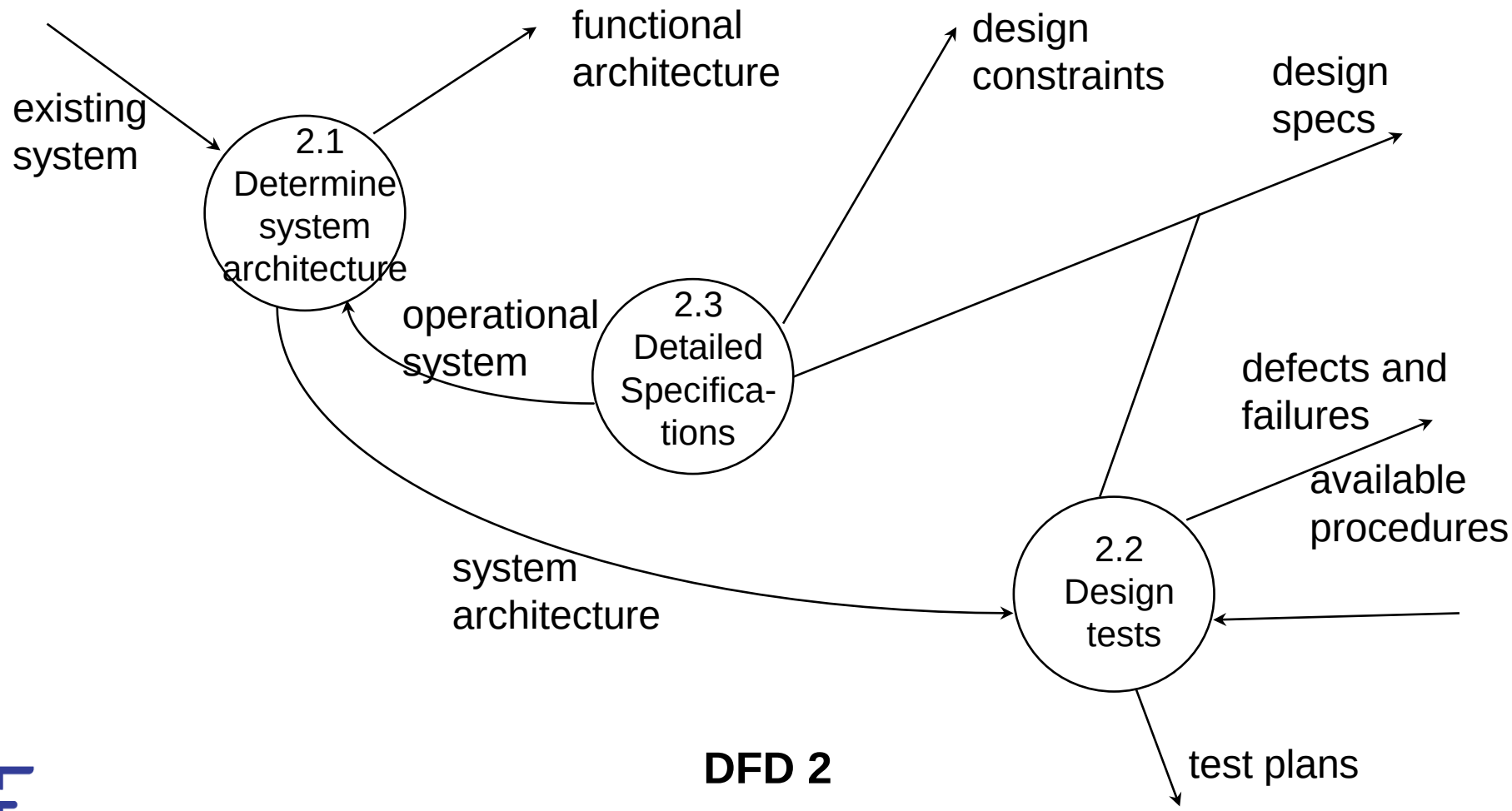
Konteks Diagram

Latihan: Cari Salah (6)



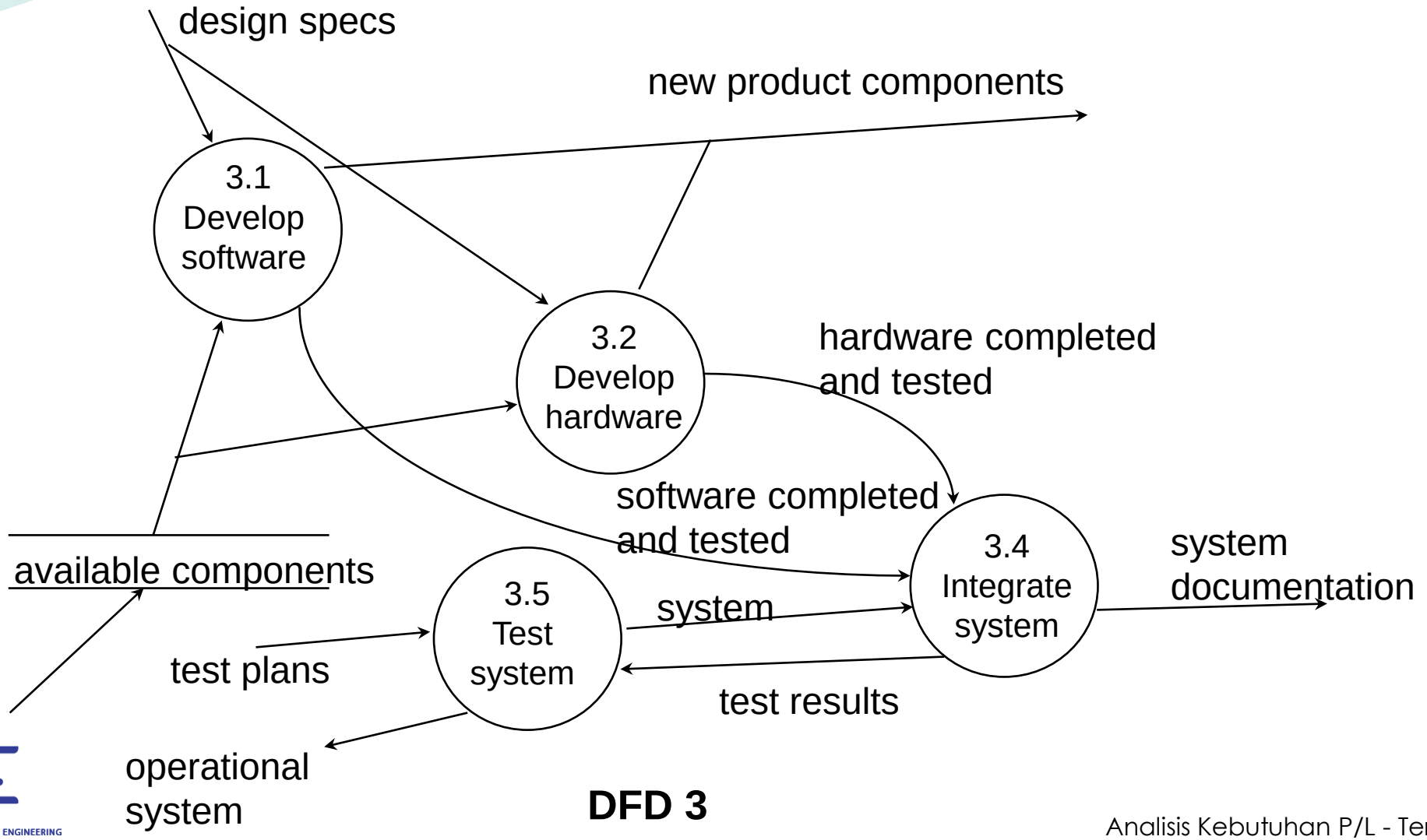
DFD level 0

Latihan: Cari Salah (7)



DFD 2

Latihan: Cari Salah (8)



DFD 3

Buat DFD: Resep Pancakes

- Seorang pelanggan bisa memesan pancake apel atau original. Pesanan pelanggan akan di kirim ke chef (tukang masak). Chef hanya dapat melayani pesanan satu pancake setiap saat. Jika pesanan pancake sudah siap, maka chef akan meletakkan ke piring dan pancake sudah siap.
- Jika kulit pancake sudah habis, maka chef akan membuat yang baru.
- Resep pancake (untuk 12 potong)
  - 1 cangkir of tepung terigu
  - Garam secukupnya
  - 1 telur
  - 1 ¼ cangkir susu
  - Sedikit minyal buat memanaskan wajan
  - 1 potongan apel

- Campurkan tepung dan garam dalam mangkok besar, tambahkan telur dan susu
- Aduk hingga halus secara hati-hati
- Panaskan wajan, masukkan 3 sendok makan mentega ke wajan, dan goyangkan wajan hingga minyak tersebar merata.
- Untuk pancake apel, tambahkan potongan apel
- Jika warnanya sudah keemasan, angkat pinggiran pancake dengan pegangan yang sesuai dan balik hingga matang sisi sebelahnya.

ANALISIS TERSTRUKTUR
State Transition Diagram

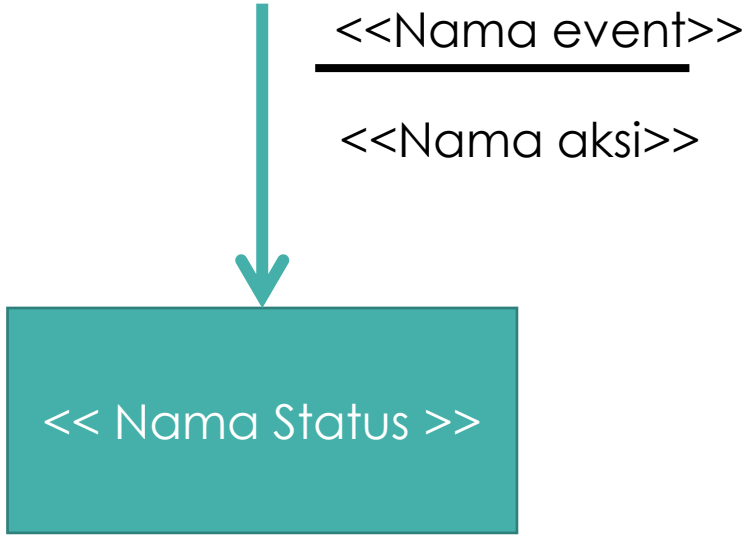
State Transition Diagram (STD) (1)

- STD menjelaskan **perilaku dinamika** sistem
  - Sistem akan memberikan **respon** terhadap suatu **stimulus** atau **event** (kejadian)
  - **Transisi** (perubahan) **status** dari **suatu entitas**
  - **Trigger** atau **event** (kejadian) yang menyebabkan **perubahan status** dari **suatu entitas**.
  - Contoh:
    - Manusia menekan keyboard (event) maka di layar tampil huruf (status)
    - Pembeli memasukkan koin (event), sistem akan menampilkan jumlah koin (status)
    - Tombol lampu di-on-kan (event), maka lampu akan memiliki status hidup dari status mati

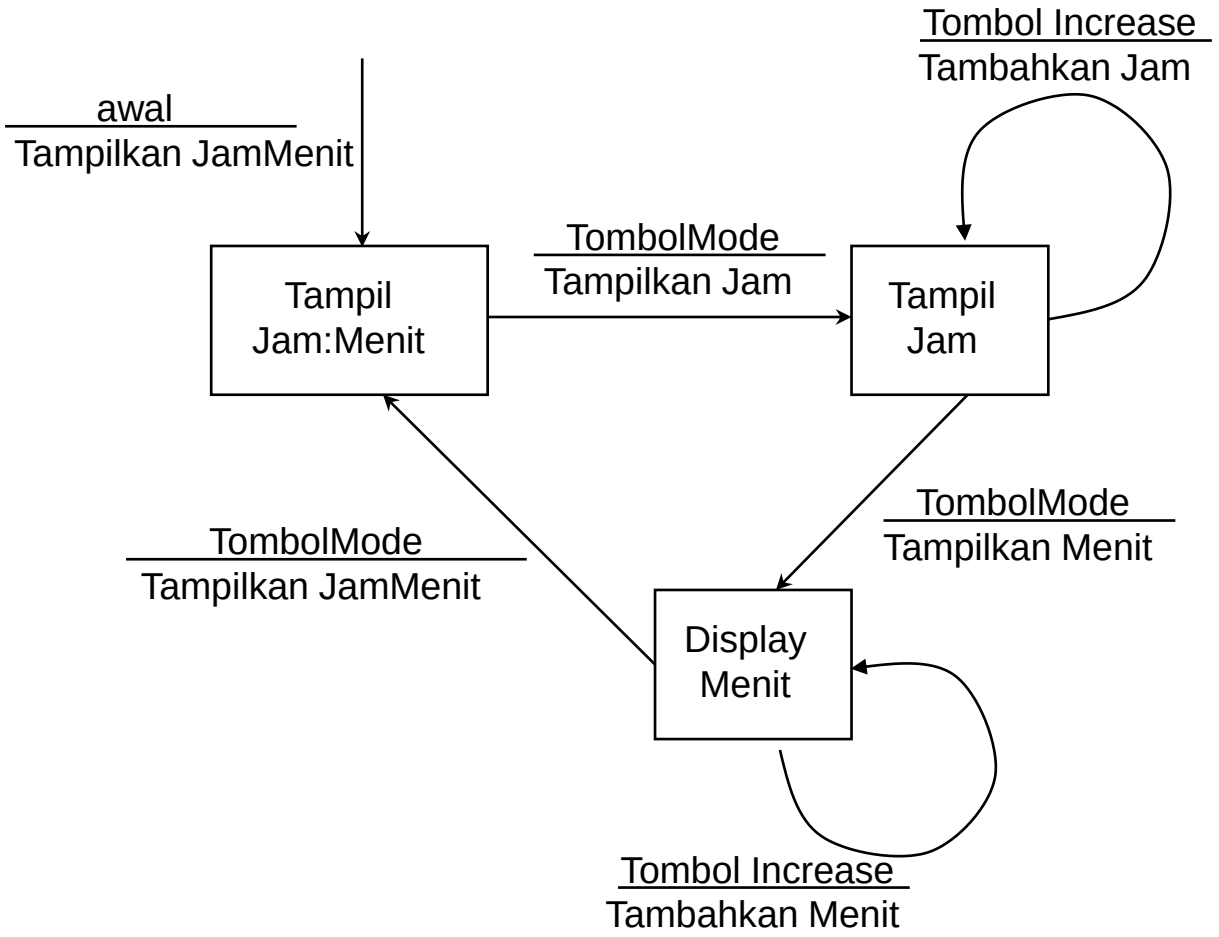
State Transition Diagram (STD) (2)

- **Event** adalah suatu **kejadian** di **luar entitas**, biasanya memerlukan suatu **aksi** untuk menjalankannya
  - **Event** dapat disebabkan karena **kemunculan suatu data** atau suatu **stimulus** karena **aktivitas manusia**.
- **Aksi** adalah apa yang **dilakukan** oleh **entitas** yang dimodelkan sebagai **respon** terhadap suatu **event**
- **Status** (state) adalah **kondisi** dari **entitas** yang dimodelkan
- **Transisi** adalah **perubahan status** karena suatu **event**, transisi digambarkan sebagai **arah panah**

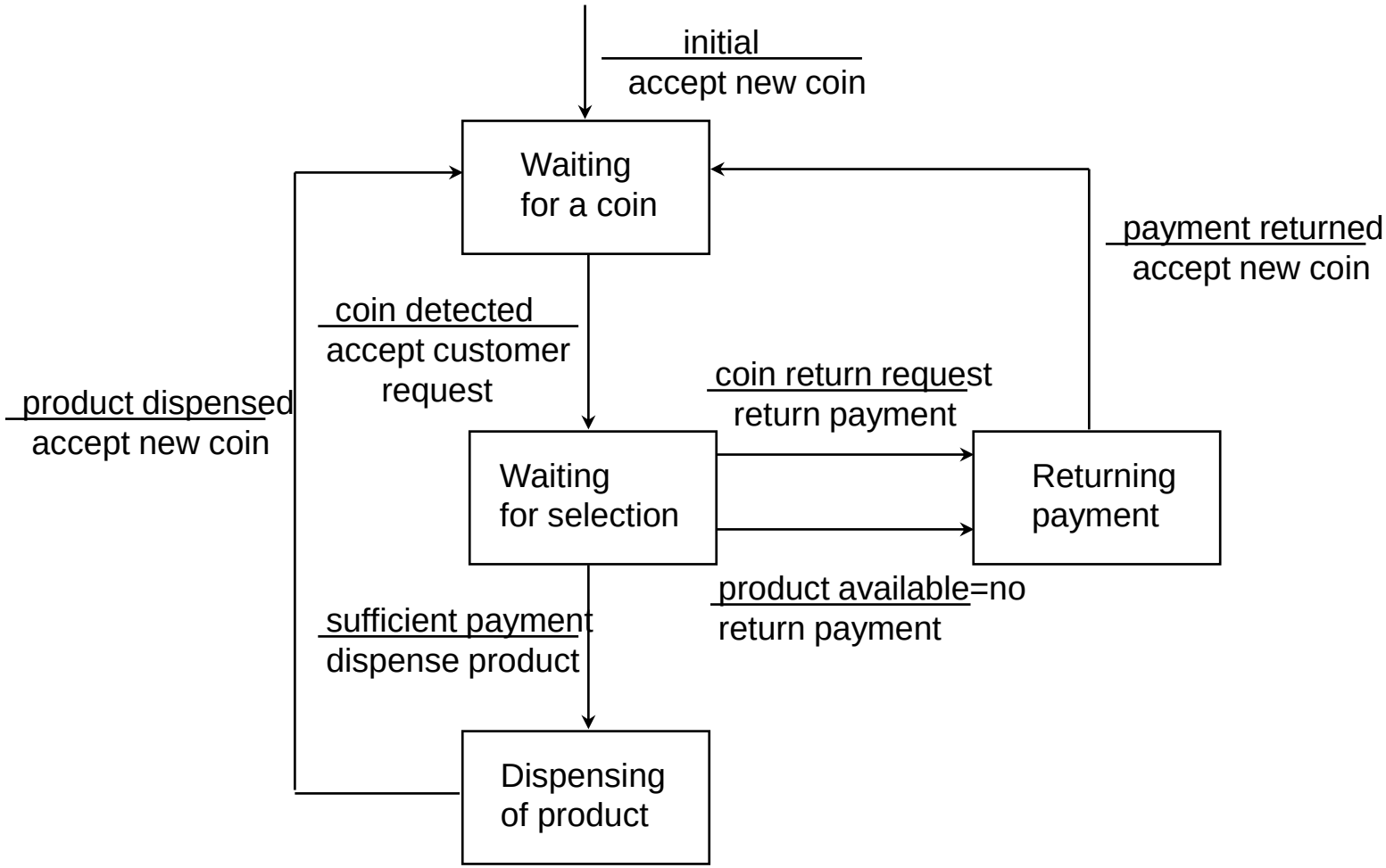
Penggambaran STD



STD untuk Jam Digital

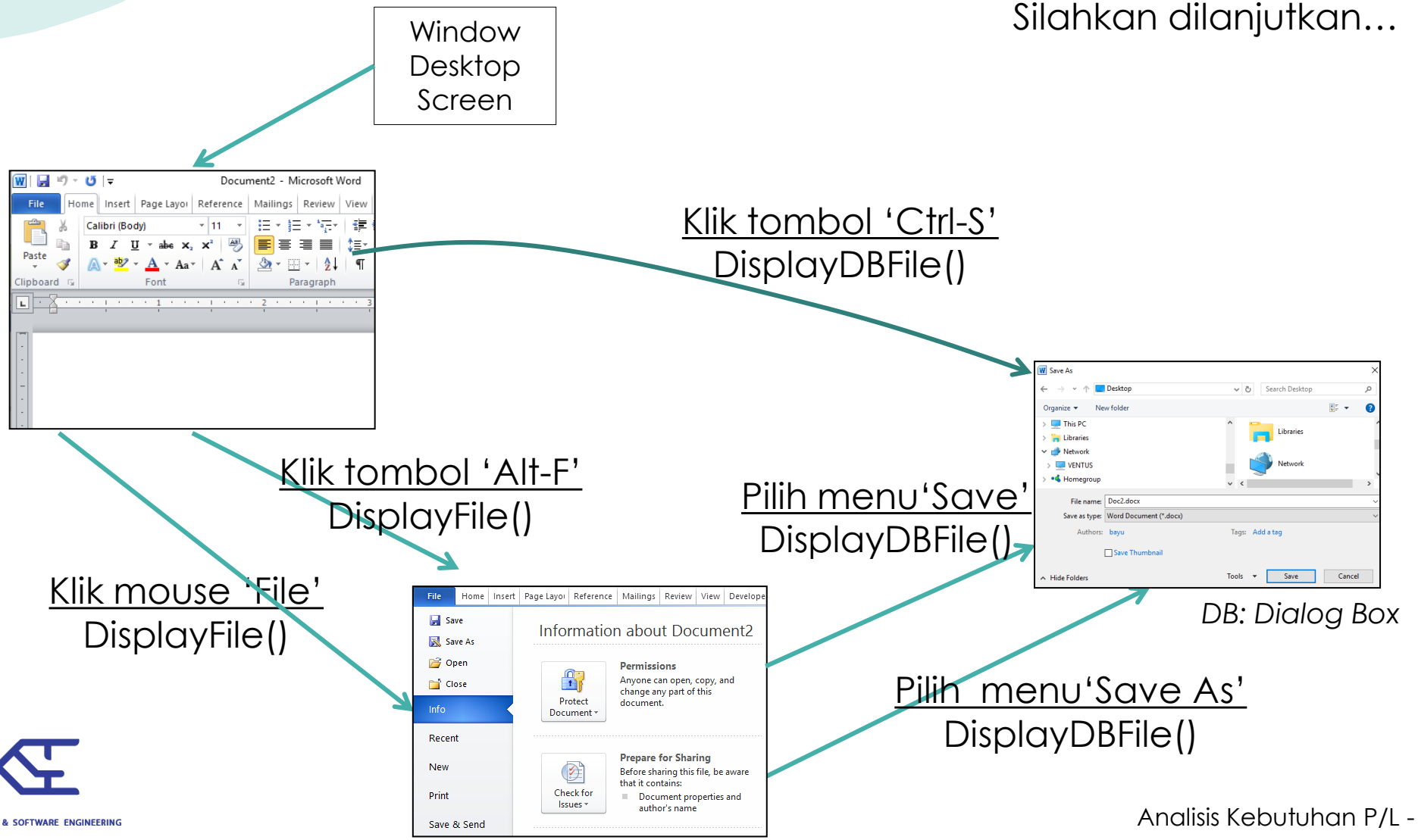


STD untuk Vending Machine <<draft>>



STD untuk Eksekusi program pada Program MS Word

Silahkan dilanjutkan…



DB: Dialog Box

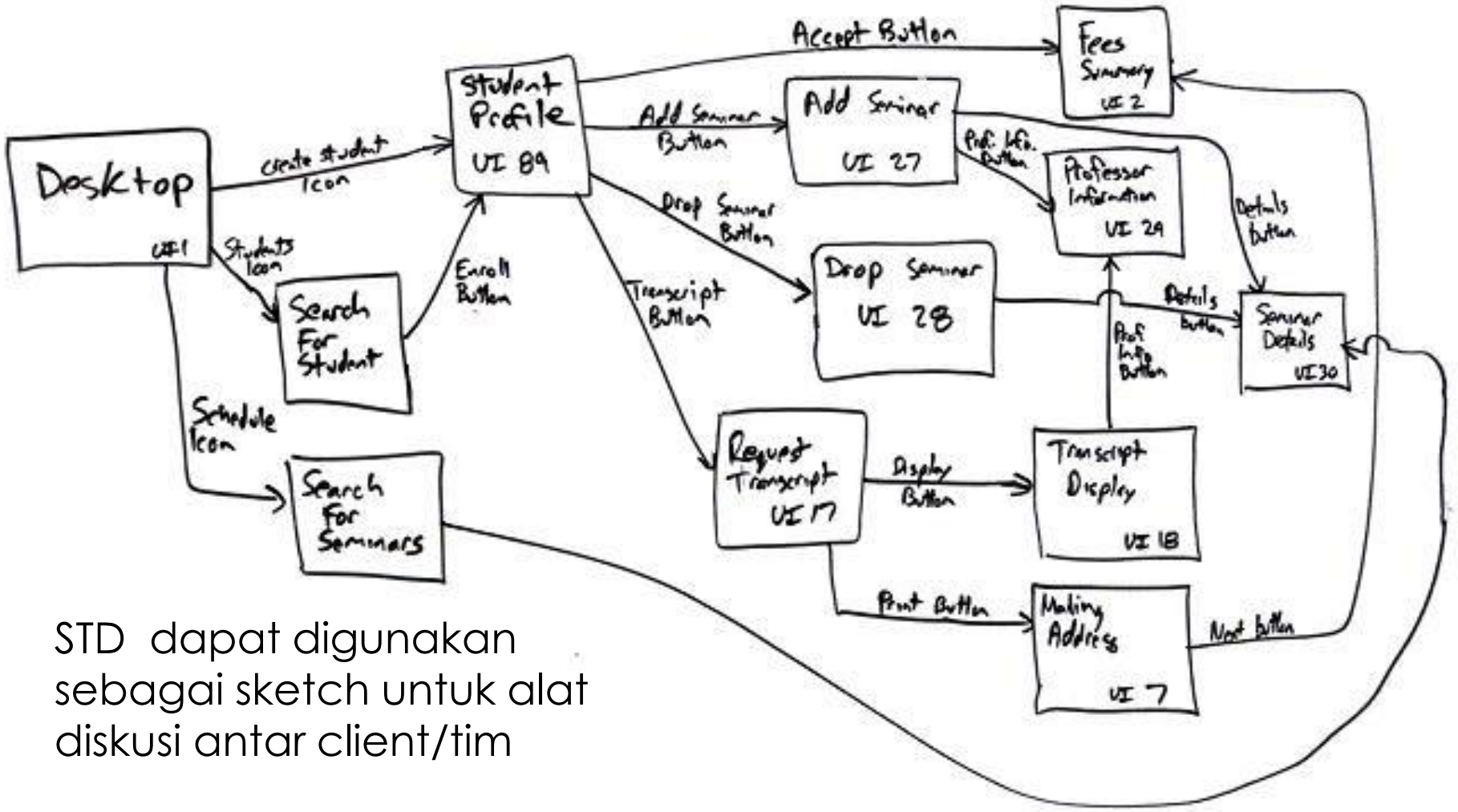
Contoh implementasi STD dalam Program

```
while (1)
{
    if (state == JamMenit) and (event == TombolMode) then
        TampilkanJam();
    if (state == Jam) and (event == TombolInc) then
        TambahkanJam();
    if (state == Jam) and (event == TombolMode) then
        TampilkanMenit();
    if (state == Menit) and (event == TombolInc) then
        TambahkanMenit();
    if (state == Menit) and (event == TombolMode) then
        TampilkanJamMenit();
    event = GetEvent();
}
```

Contoh implementasi STD dalam Program

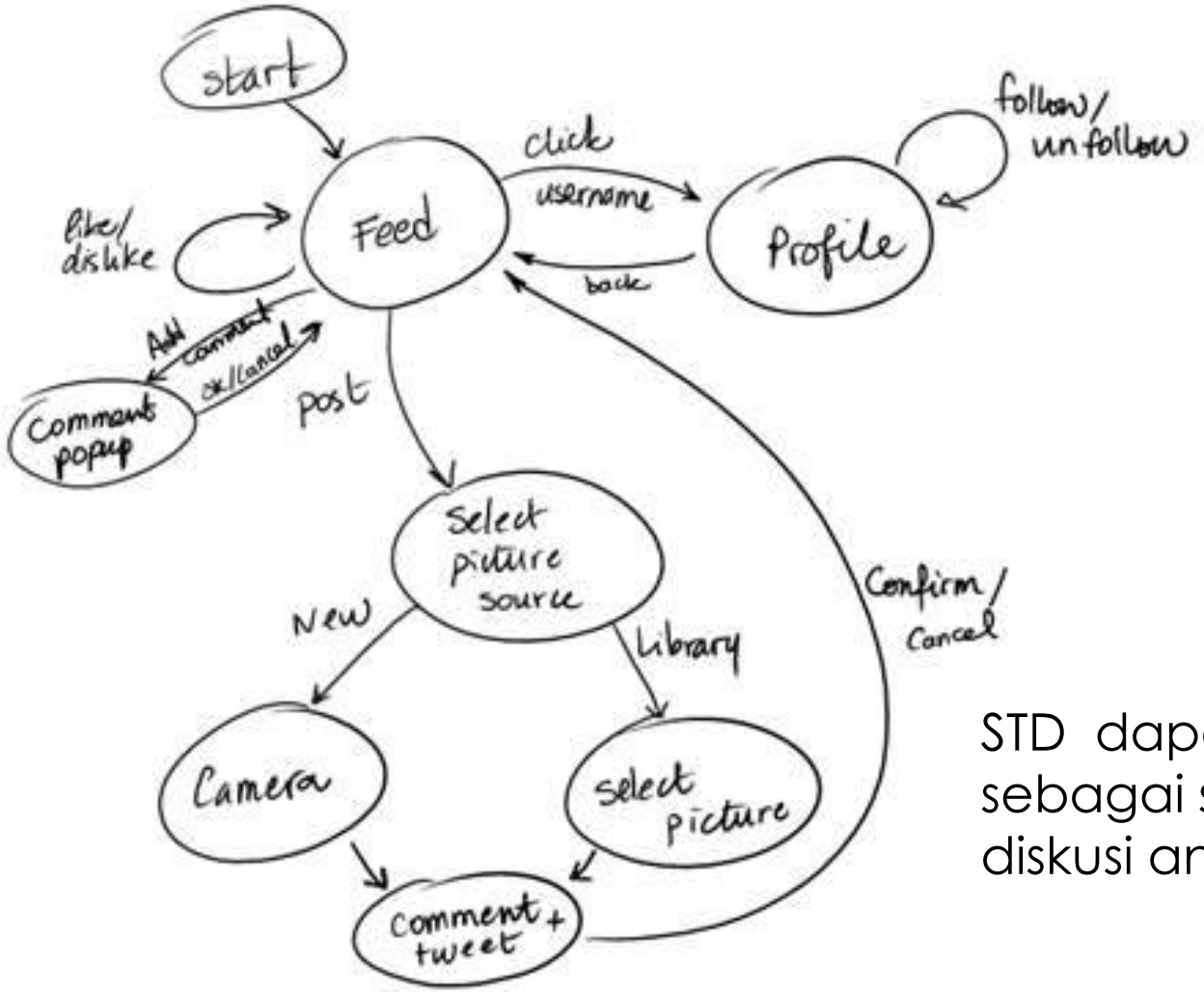
```
while (1)
{
    switch (state)
        jamMenit: switch (event)
                    TombolMode: TampilkanJam();
                    TombolInc  : -
                  end;

        jam :      switch (event)
                TombolMode: TampilkanMenit();
                     TombolInc:  TambahkanJam();
                  end;
        menit:     switch (event)
                TombolMode: TampilkanJamMenit();
                     TombolInc:  TambahkanMenit();
                  end;
    end;
    event = GetEvent();
}
```

Varian Implementasi STD (1)



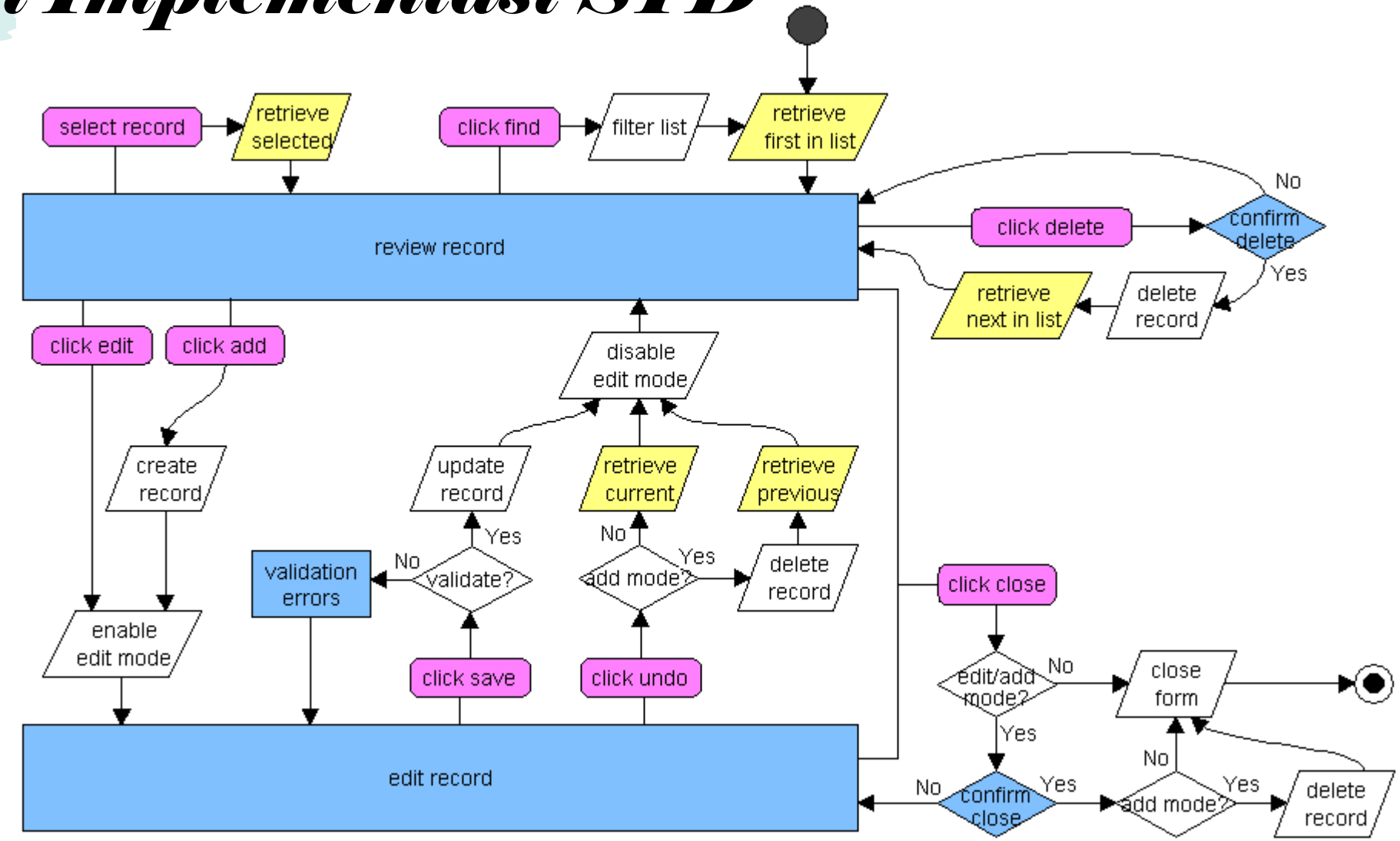
STD dapat digunakan sebagai sketch untuk alat diskusi antar client/tim

Varian Implementasi STD (2)

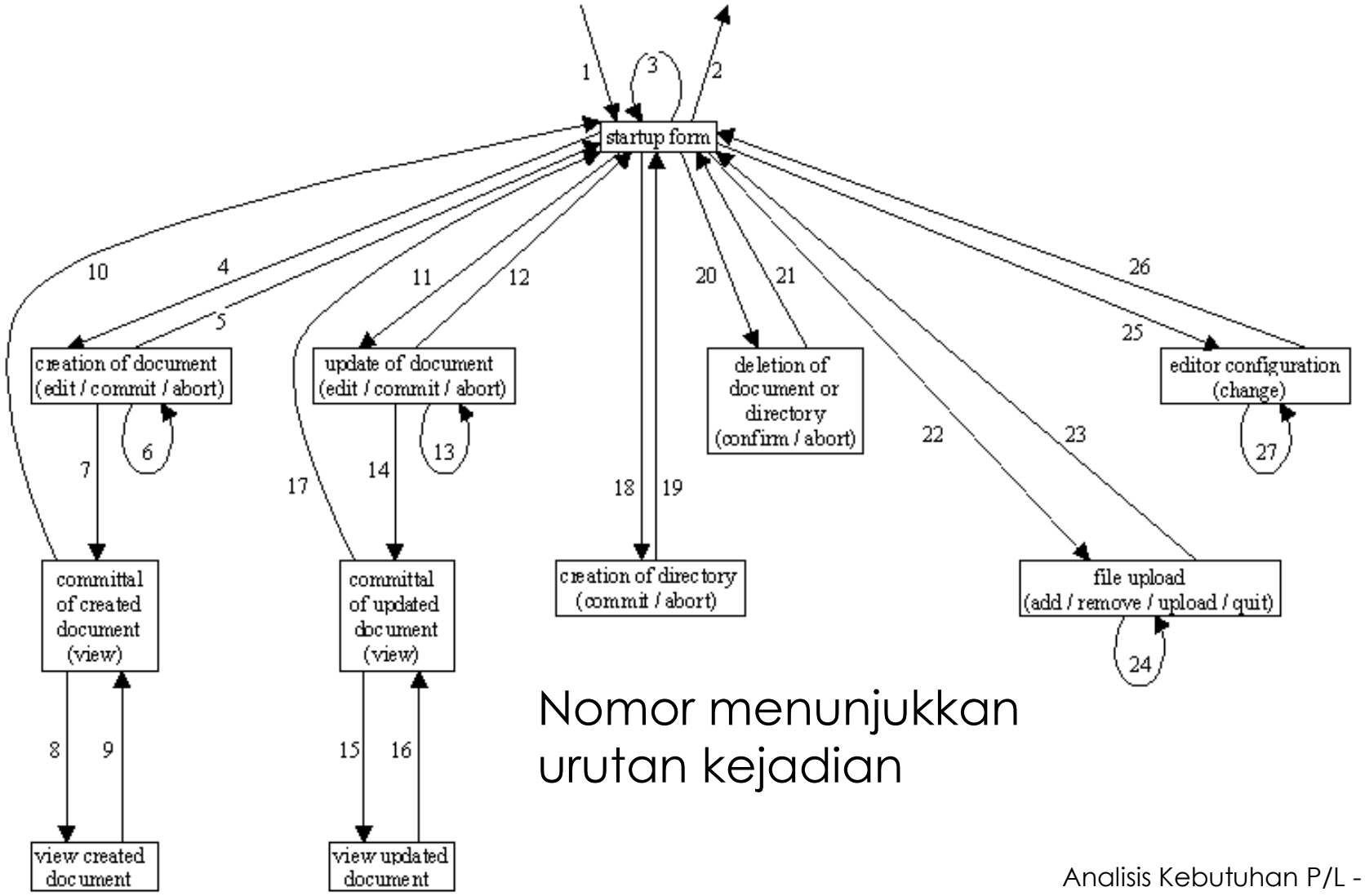


STD dapat digunakan sebagai sketch untuk alat diskusi antar client/tim

Varian Implementasi STD

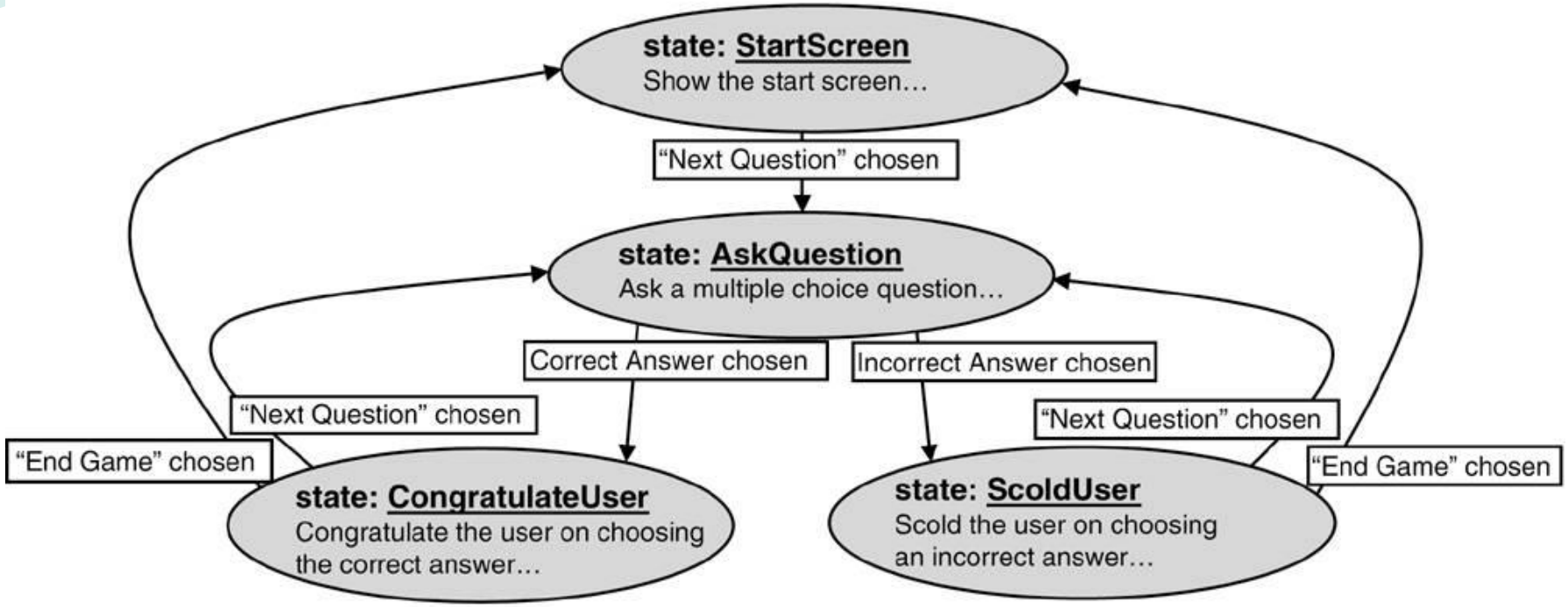


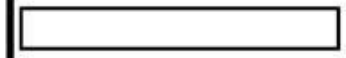
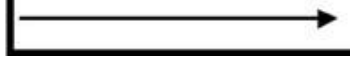
Varian Implementasi STD

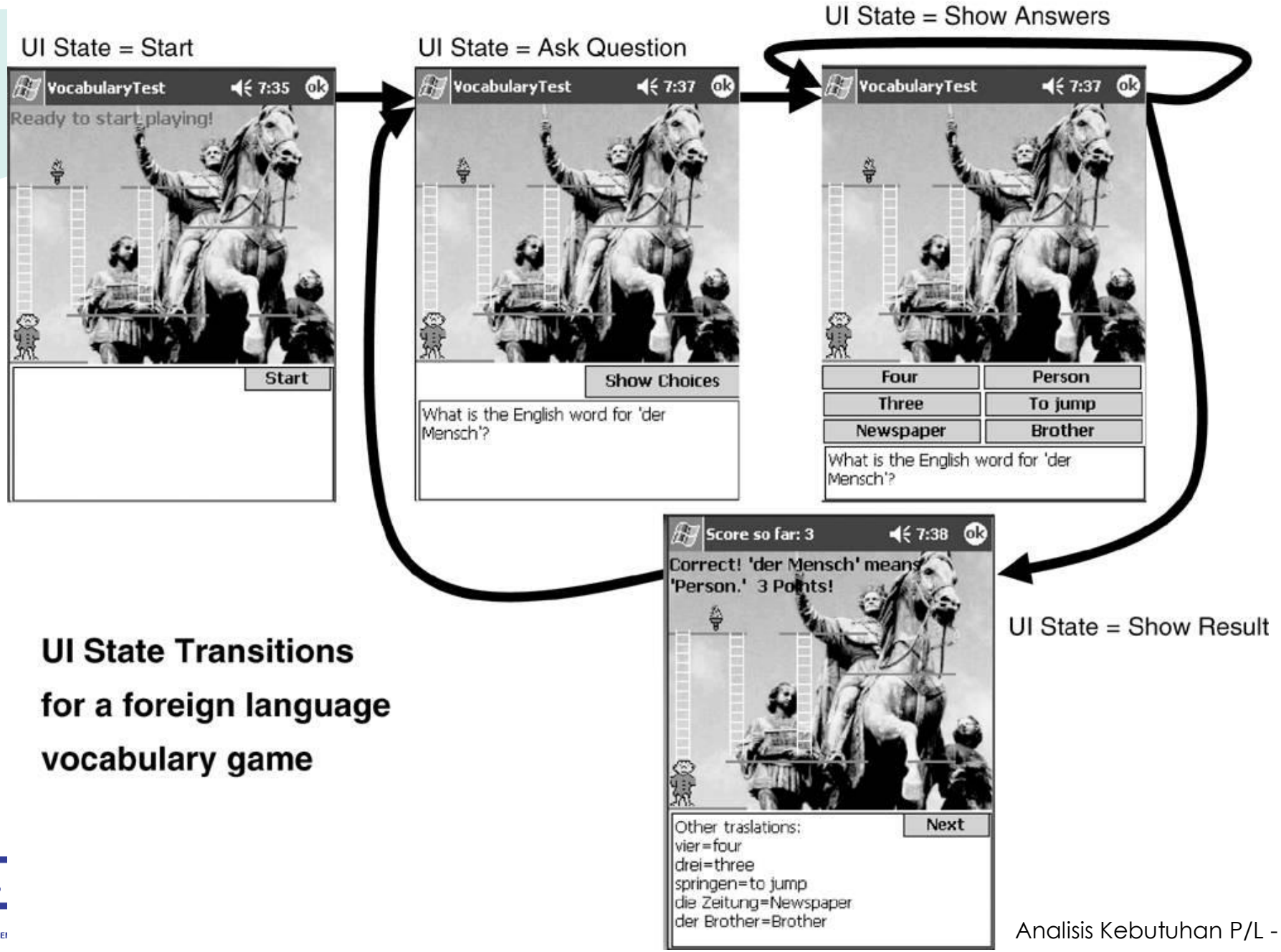


Nomor menunjukkan urutan kejadian

State Machine for simple multiple choice game



Item	Meaning
	State
	External Input
	Transition Logic



UI State Transitions for a foreign language vocabulary game

Rangkuman

- DFD/ERD/STD adalah **abstraksi** model dari **dunia nyata**
  - **DFD** menggambarkan **proses** dan **aliran data** yang **berpindah** antar **proses**
  - **ERD** menggambarkan **hubungan** antar entitas
  - **STD** menggambarkan **urutan** perubahan **status** secara **dinamis** bila diberikan suatu **stimulus** (event)

Rangkuman (2)

- DFD/ERD/STD adalah **alat diskus**i antara **pengembang** dengan **user/customer**
  - Bentuk diagram ini umumnya lebih mudah dimengerti oleh user/customer
  - Bagi pengembang diagram ini juga membantu merancang dan mengimplementasikan solusi
  - Bentuk diagram ini **berevolusi** sesuai dengan aktivitas **pengumpulan kebutuhan**
    - Hasil final biasanya berbeda dengan diagram awal

Rangkuman

Iterasi, iterasi, iterasi!!

- Jangan takut untuk mulai dari awal
  - Sering terjadi ketika menyiapkan DFD di level bawah, maka induknya ikut berubah
- Kita tidak akan mengerti proses lengkap sebelum kita selesai
  - DFD akan menjadi jelas setelah diagram selesai di gambar