Tim Pengajar IF2250

IF2250 – Rekayasa Perangkat Lunak

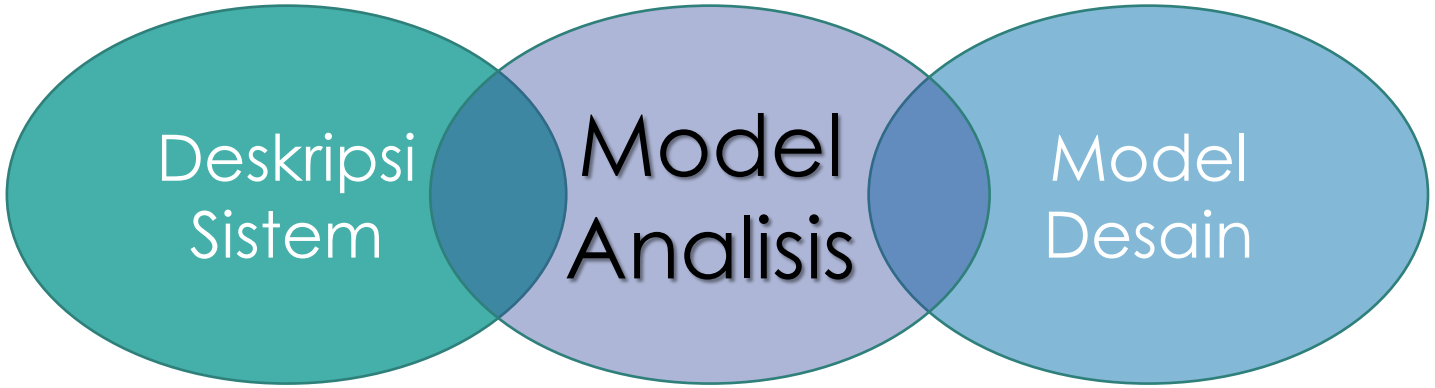
Scenario-based Modeling

SEMESTER II TAHUN AJARAN 2023/2024

KNOWLEDGE & SOFTWARE ENGINEERING

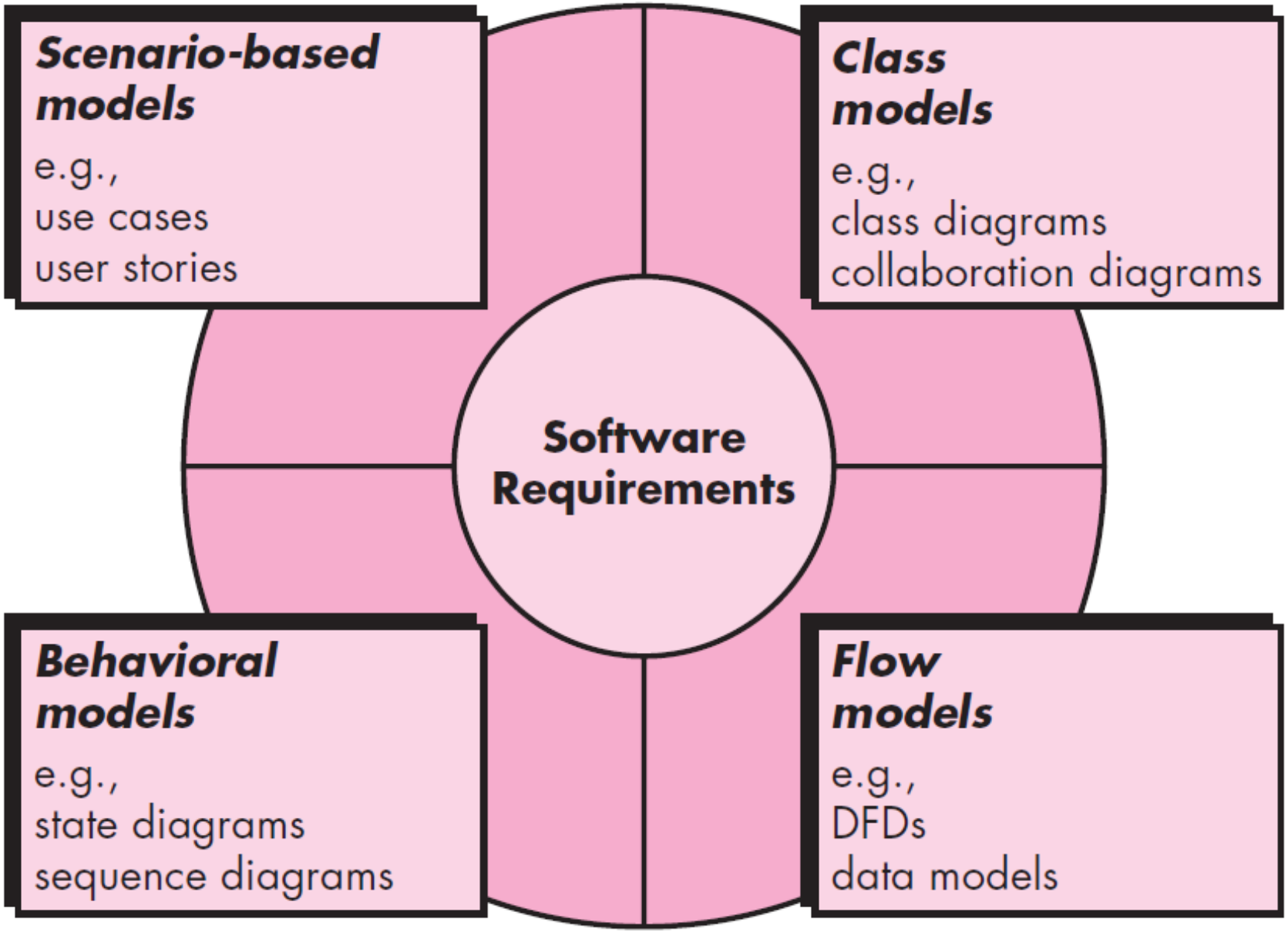
Model Analisis

Model Analisis adalah "jembatan" antara Deskripsi Sistem dan Model Desain



- Model harus difokuskan pada kebutuhan yang ada dalam domain masalah/bisnis
- Tiap elemen digunakan untuk mengerti kebutuhan sistem
- Model dibuat sederhana

KNOWLEDGE & SOFTWARE ENGINEERING


Scenario-based models
e.g.,
use cases
user stories
Class models
e.g.,
class diagrams
collaboration diagrams
Software Requirements
Behavioral models
e.g.,
state diagrams
sequence diagrams
Flow models
e.g.,
DFDs
data models

Model Analisis Kebutuhan PL

- Model Berorientasi Aliran (***Flow-Oriented Models***)
  - Contoh: DFD
- Model Perilaku (***Behavioral Models***)
  - Contoh: *State Transition Diagram* atau *State Diagram*
- Model Berbasis Skenario (***Scenario-Based Models***)
  - Fungsional – pemrosesan cerita untuk suatu fungsi perangkat lunak
  - *Use-Case* – deskripsi hubungan interaksi antara aktor dan sistem
- Model Berbasis Kelas (***Class-Based Models***)
  - Dikembangkan berdasarkan entitas yang ditemukan saat melakukan pemodelan skenario dan/atau fungsional

Pemodelan Berbasis Skenario (Scenario Based Modeling)

Scenario-based Modeling

- Apa yang dimodelkan?
  - Cara pengguna **berinteraksi** dengan sistem
- Seperti apa modelnya?
  - Diagram *use-case*
  - Diagram aktivitas
  - Diagram aktivitas dalam bentuk *swimlane*
  - Diagram interaksi
- Bagaimana membuat modelnya?

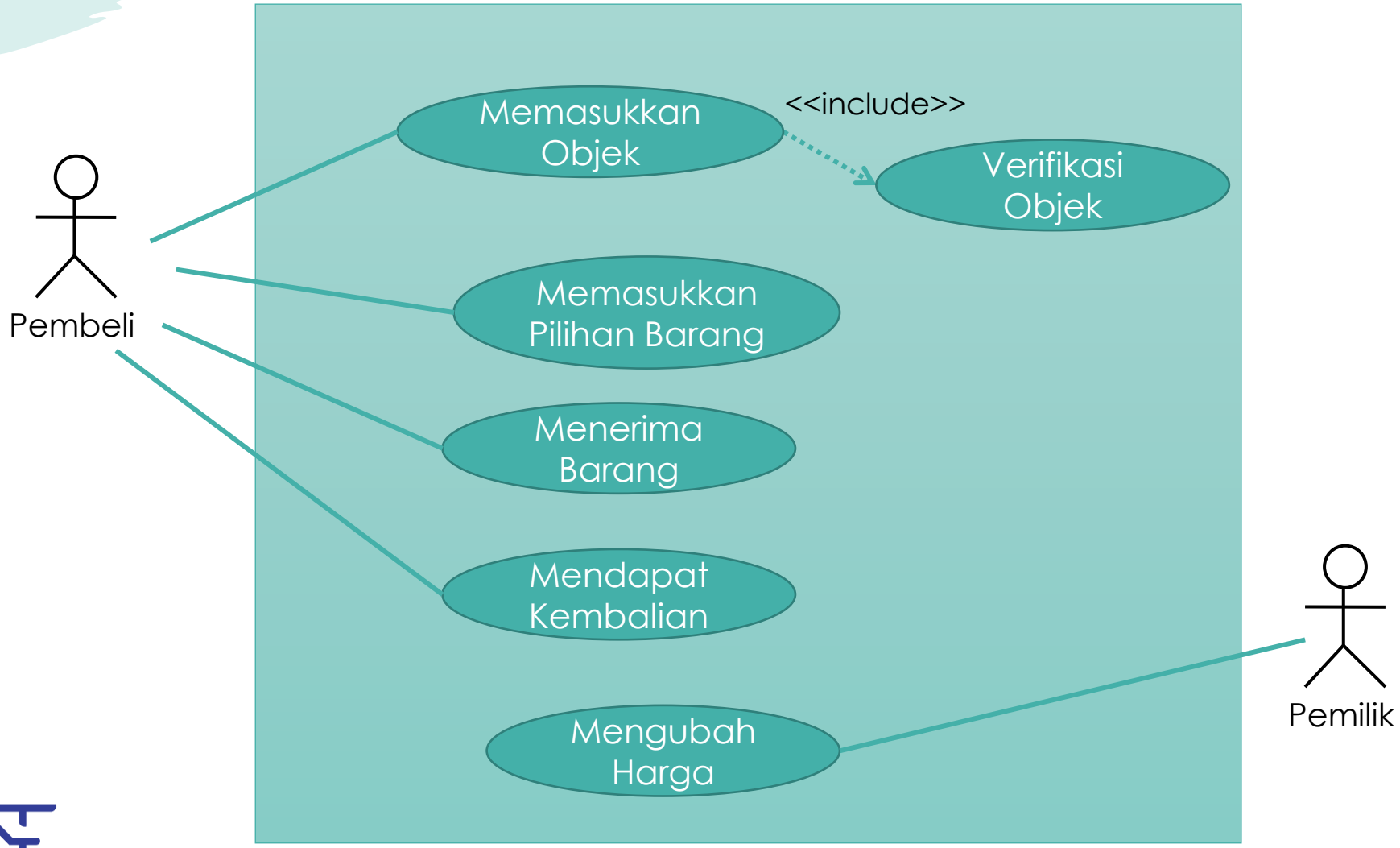
Diagram Use-case

- Apa itu *use-case*?
- Apa itu aktor?
- Apa itu skenario?

Ivar Jacobson: "*[**Use-cases**] are simply an aid to defining what exists outside the system (**actors**) and what should be performed by the system (**use-cases**).*"

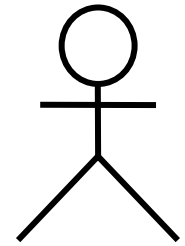
KNOWLEDGE & SOFTWARE ENGINEERING

Use Case untuk VM

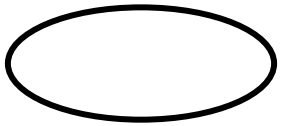


Konsep Pemodelan use-case

- Aktor mewakili semua yang **berinteraksi** dengan sistem
  - Aktor adalah unsur '**eksternal**'
- *Use-case* adalah ***urutan aksi-aksi*** dalam sistem yang melakukan suatu pekerjaan yang memberikan suatu hasil untuk aktor
  - use-case bertindak sebagai **penghubung** antara pengguna dengan pengembang
  - use-case berguna sebagai **alat komunikasi** antara pengguna dan pengembang



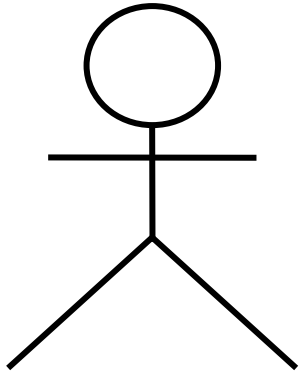
Actor



Use-Case

Aktor (Actor)

- Aktor bukan bagian dari sistem
  - Mereka berperan sebagai 'pengguna' dari sistem
- Aktor mungkin secara aktif bertukar informasi dengan sistem
- Aktor mungkin berfungsi pasif sebagai penerima informasi
- Aktor bisa merepresentasikan
  - Manusia,
  - Mesin,
  - Sistem lain



Actor

Aktor mewakili unsur Eksternal

KNOWLEDGE & SOFTWARE ENGINEERING

Seorang pengguna mungkin memiliki peran berbeda

Ali

Ali sebagai mahasiswa

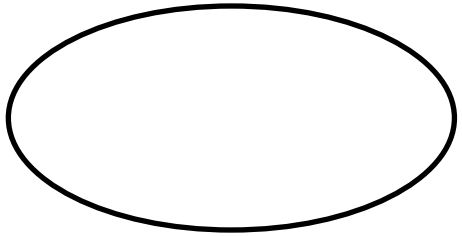
Mahasiswa

Ali sebagai Dosen

Dosen

Dengan *use-case* peran ini harus digambarkan berbeda walau dalam kenyataannya orangnya mungkin hanya satu

Use-Case



Use-Case

- *Use-case* merepresentasikan **dialog** antara aktor dengan sistem
  - *use-case* di'**inisiasi**' oleh **aktor** untuk melakukan suatu fungsi tertentu dalam sistem
- *Use-case* merepresentasikan dialog antara satu atau lebih aktor
  - sistem akan **mengembalikan** suatu **nilai** ke aktor
- *Use-case* perlu menggambarkan ***event*** yang lengkap dan memiliki makna bagi sistem
- *Use case* juga dapat dilihat sebagai tujuan sistem secara umum yang mungkin melibatkan satu atau lebih aktor
- Semua *use-case* mengarahkan ke semua penggunaan sistem

Penjelasan Use-case

- Gambar *use-case* (termasuk aktornya) perlu disertai dengan **keterangan** yang akan membantu **menjelaskan** gambaran yang diberikan
  - Keterangan ini tidak perlu panjang lebar, untuk setiap *use-case* lebih kurang **dua baris** saja.
  - Deskripsi langkah-demi langkah apa yang perlu dikerjakan sistem ketika berinteraksi dengan aktor
- Perlu dijelaskan bagaimana antar *use-case* **saling terkait**
  - Atau suatu *use case* yang terlibat pada suatu *use-case* lain
  - Bagaimana *use case* itu dilakukan oleh aktor
  - *Use case* dapat dikelompokkan sebagai paket (*package*)

Isi Rincian dari Use-case

- Deskripsi struktur *Use-case*
  - Menyatakan bahwa instansiasi dari *use-case* akan memiliki perubahan **transisi antar *state*** (status)
    - Perubahan ini digambarkan dalam aliran *event* (*event flow*)
  - Setiap transisi adalah sekumpulan aksi
    - Mungkin bisa makin komplek, jadi penjelasan sesederhana mungkin
- Deskripsi *Use-case* bisa dimulai dari ***Basic Path***
  - **Alternatif path** diberikan penjelasan pada bagian yang berbeda
- Berikan penjelasan yang mudah dibaca

Apa saja isi deskripsi

- Penjelasan **Kondisi awal**
- Bagaimana dan kapan *use-case* **dimulai**
  - Urutan kejadian (aliran *event*)
  - Bagaimana dan kapan *use-case* selesai
  - Kemungkinan *post*-kondisi alternatif
  - Jalur eksekusi yang tidak diperbolehkan
  - Jalur alternatif (diambil dari jalur dasar)
- Interaksi sistem dengan aktor
- Penggunaan objek, sumber daya dalam sistem
- Penjelasan apa yang dilakukan oleh sistem

Deskripsi Formal

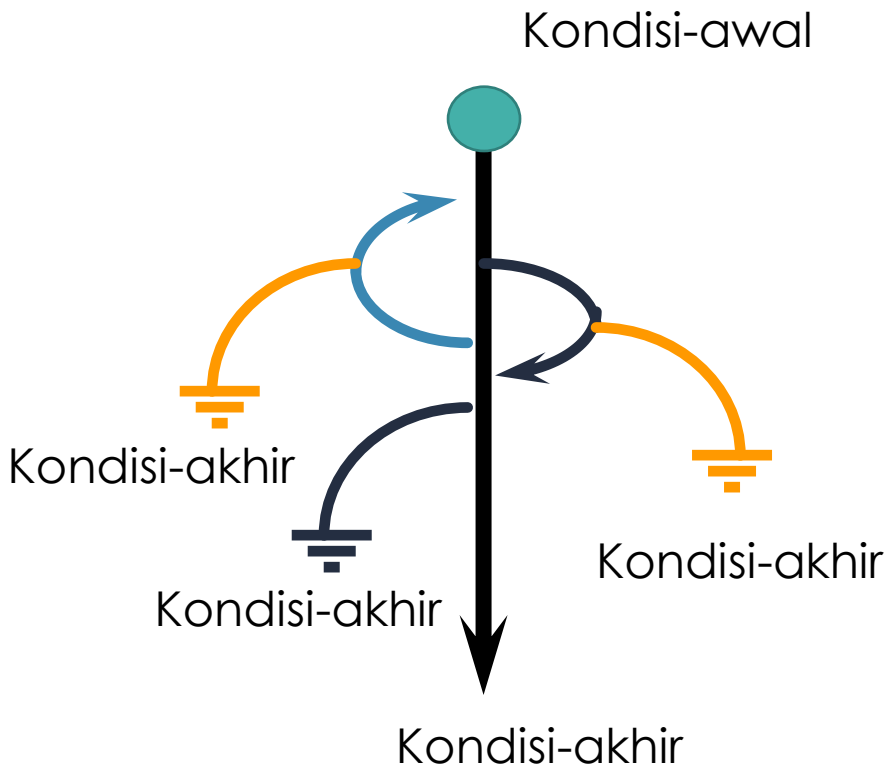
- Untuk jumlah *use-case* yang **besar** dengan berbagai **alternatif**, maka penulisan teks menjadi tidak praktis, jadi kadang digunakan **diagram**
  - **Diagram *Statechart***
    - Untuk menggambarkan *use-case* yang kompleks
    - Berisi penjelasan *state* dan transisi dalam *use-case*
  - **Diagram Aktivitas**
    - Menggambarkan transisi antar *state* dalam bentuk urutan aksi
    - Bentuk yang lebih umum dari *State Transition Diagram*
  - **Diagram Interaksi**
    - Menjelaskan interaksi antar instansiasi dari aktor dan instansiasi dari *use-case*

Aliran Event (Flow of Event) (1)

- Aliran ini bisa bersifat
  - Aliran yang '**normal**', atau aliran 'dasar' atau jalur dasar (*basic path*) atau "*Happy path*"
  - Aliran **alternatif**
    - Varian yang regular
    - Penanganan kasus khusus
    - Aliran yang khusus menangani terjadinya '*error*'

Contoh:

- Basic Path untuk *use case* 'Terima Barang' adalah ketika pelanggan memasukkan suatu item barang dan meminta tanda terima
- Alternatif path adalah *use case* ketika suatu barang terjepit di slot pemasukan barang



Aliran Event (Flow of Event) (2)

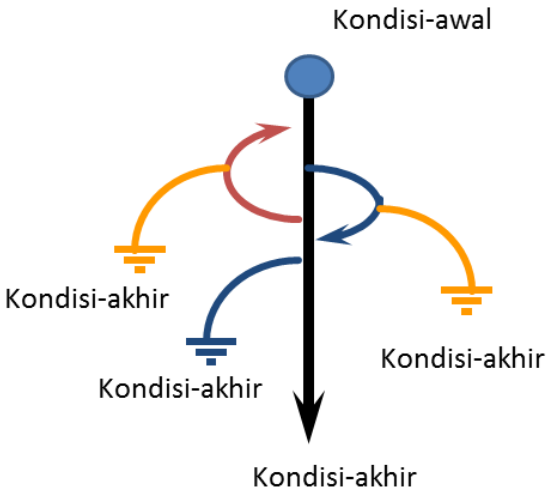
- Aliran Basic (***basic flow***) atau **normal** (*flow*) menjelaskan
  - Apa yang terjadi pada '**umum**'nya kalau suatu *use-case* dilakukan
  - Sering disebut sebagai '***happy path***' atau skenario '*happy path*'
- Aliran *event* dalam *use-case* bisa dibagi menjadi beberapa '***sub flow***'
  - Beberapa bagian dari *flow event* bisa dipecah dan pecahannya dapat diberikan penjelasan secara terpisah; untuk meningkatkan kemudahan pembacaan (*readability*), dan menjadi perbaikan dari struktur model *use-case*
  - Pemisahan ini juga membuat *flow* dasar menjadi lebih jelas
  - Jika suatu *subflow* melibatkan hanya sebagian kecil dari *event flow* lengkapnya, mungkin cukup dijelaskan dalam teks saja.
  - *Sub flow* kadang sebaiknya dipisahkan sebagai suplemen terpisah dalri bagian *flow of events*

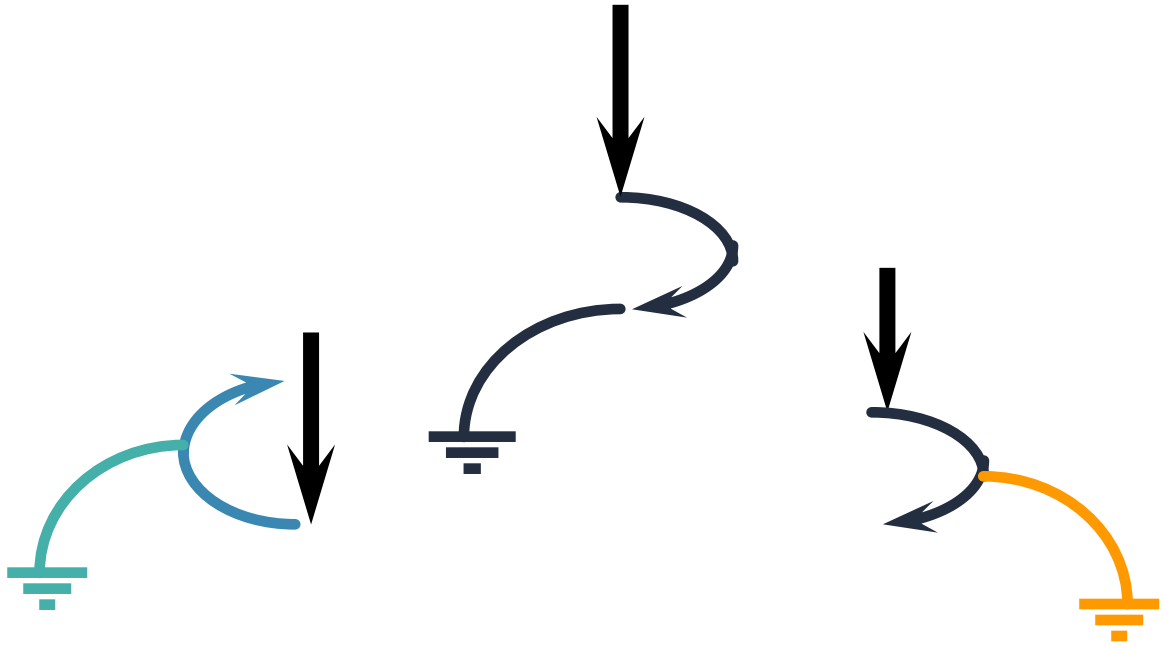
Aliran Event (Flow of Event) (3)

- *Use-case* harus mencakup **semua flow**, baik yang **normal**, **alternatif** maupun *flow* pengecualian (***exceptional***).
  - *use-case* harus digambarkan sedemikian rupa sehingga mudah untuk melihat *flow* dan mudah dimengerti apa yang terjadi pada satu saat
- Contoh *subflow*
  - Bila menempati bagian yang cukup besar dalam *event flow*
  - Jika merupakan *event flow* varian, atau *exception*
  - Bisa dieksekusi pada beberapa interval yang dalam *event flow* yang sama
  - *Sub flow* mungkin dilakukan pada waktu yang sama, dan mungkin bersifat *optional*

Skenario

- Skenario adalah hasil 'instansiasi' dari *use-case*
  - Berisi satu aliran (***flow***) dalam suatu *use-case*





Garis hitam tebal merepresentasikan skenario yang mungkin untuk aliran basic dan alternatif

Skenario (2)

- Setiap *use-case* akan memiliki sekumpulan *flow-event* dengan skenario sebagai bagian dari instansiasi suatu aliran *event*
  - Skenario mungkin melibatkan *basic flow* dan alternatif *flow* (dalam berbagai kombinasi)
- Berapa skenario yang dibutuhkan?
  - Sebanyak mungkin, agar sistem dapat dimengerti hingga mudah dikembangkan
  - Dan perlu ditekankan jika ada *use-case* yang 'menarik' atau memiliki resiko tinggi.
    - Skenario dapat digunakan untuk mengerti, juga untuk mem-validasi aliran *event* dalam *use-case*
- Dalam pemodelannya, ada yang menulis skenario lalu *use-case* atau sebaliknya
- Skenario dapat memudahkan pengembangan kasus pengujian (*test case*)

Langkah pemodelan berbasis skenario

- Cari aktor
- Cari *use case*
- Gambarkan diagram *use case*
- Buat skenario tiap *use case*

Cari aktor

- Siapa '**pengguna**' sistem atau yang terkait dengan sistem
  - Pelanggan
  - Operator
- Pada kasus ini,
  - Ada aktor yang **menggunaka**n sistem
  - Ada aktor yang bertugas melakukan perawatan (***maintenance***)
- **Peran** aktor harus **berbeda**
  - Mungkin bisa terjadi peran yang saling tumpang tindih
    - Perlu identifikasi yang jelas (mungkin melibatkan diskusi panjang dengan calon pemilik sistem)
  - Perlu **nama** yang 'relevan' dengan makna semantik dari peran
    - Contoh:
      - **Pengguna sistem** dengan **pelanggan sistem**, harus jelas apa yang dimaksud dengan peran ini. Misalnya pengguna ATM adalah juga pelanggan (*customer*) dari Bank. Gunakan istilah yang jelas perannya, bila kurang jelas, perlu diperjelas dalam penjelasan deskripsinya.
        - Istilah pengguna sistem mungkin juga adalah sistem lain (bukan orang).
  - Juga harus jelas apa '**kebutuhan**' dan '**tanggung jawab**' si aktor!

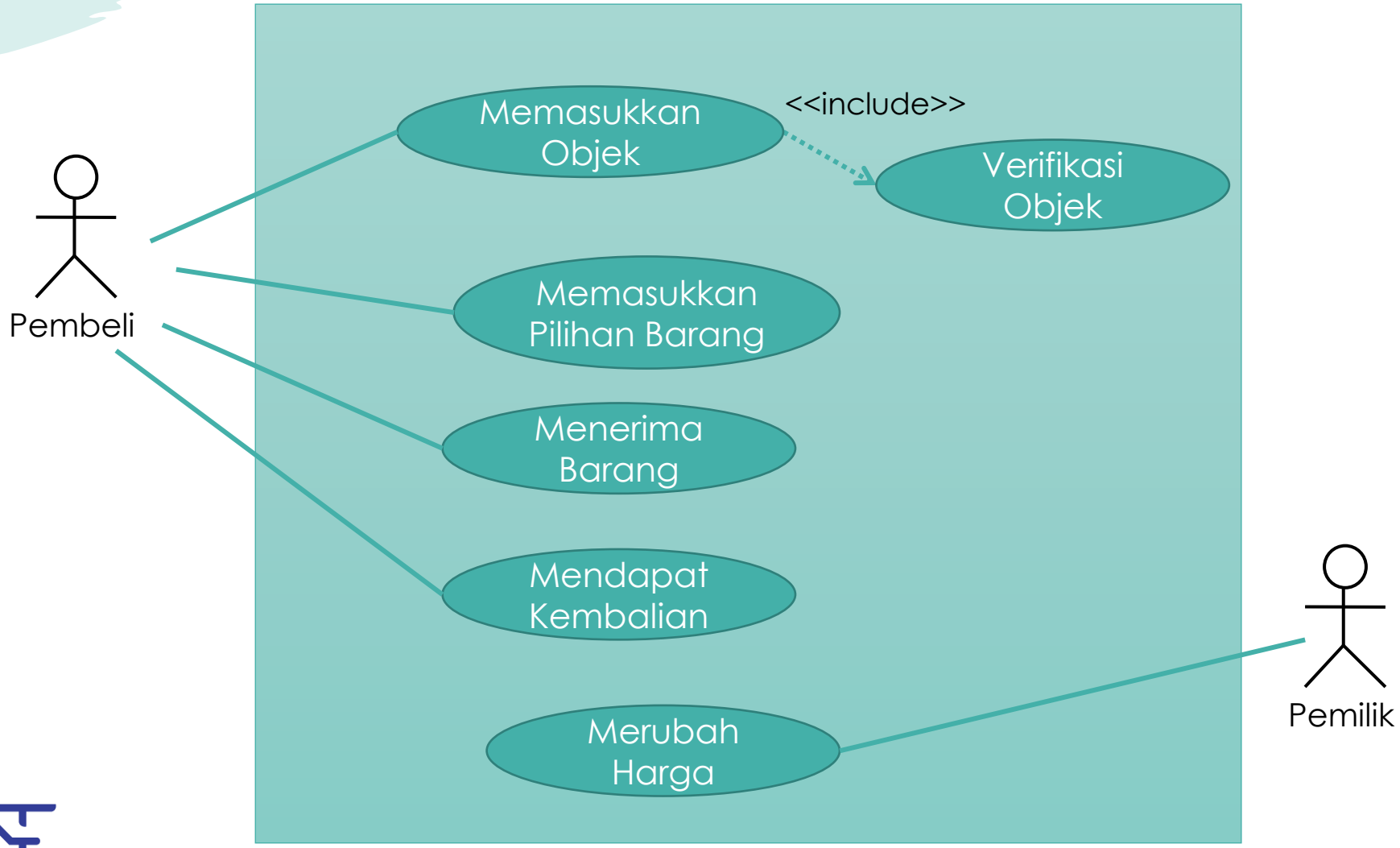
Cari use-case

- Daftarkan **aktivitas** yang dilakukan oleh **aktor** untuk melakukan suatu **fungsi/kegiatan**
  - Lakukan untuk semua aktor
- Beri **nama**
  - Nama ini sebaiknya membuat kita berpikir terhadap sekumpulan aksi yang akan terkait dengan aktor
  - Biasanya dimulai dengan '**kata kerja**'
- *Use-case* biasanya harus '**lengkap**' atau dapat '**berdiri sendiri**'
- *Use-case* ini memberikan suatu '**hasil**' untuk **aktor** ini
- *Use-case* biasanya ditulis dalam bentuk **cerita** yang kemudian dipetakan ke suatu template
- Setiap skenario utama harus dikaji ulang (*review*) dan diperbaiki untuk melihat alternatif interaksi yang mungkin
  - Apakah aktor bisa punya peran lain atau melakukan aksi lain
  - Apakah mungkin aktor mengalami kondisi '*error*'
  - Apakah mungkin aktor melakukan perilaku/aksi yang berbeda pada suatu titik?

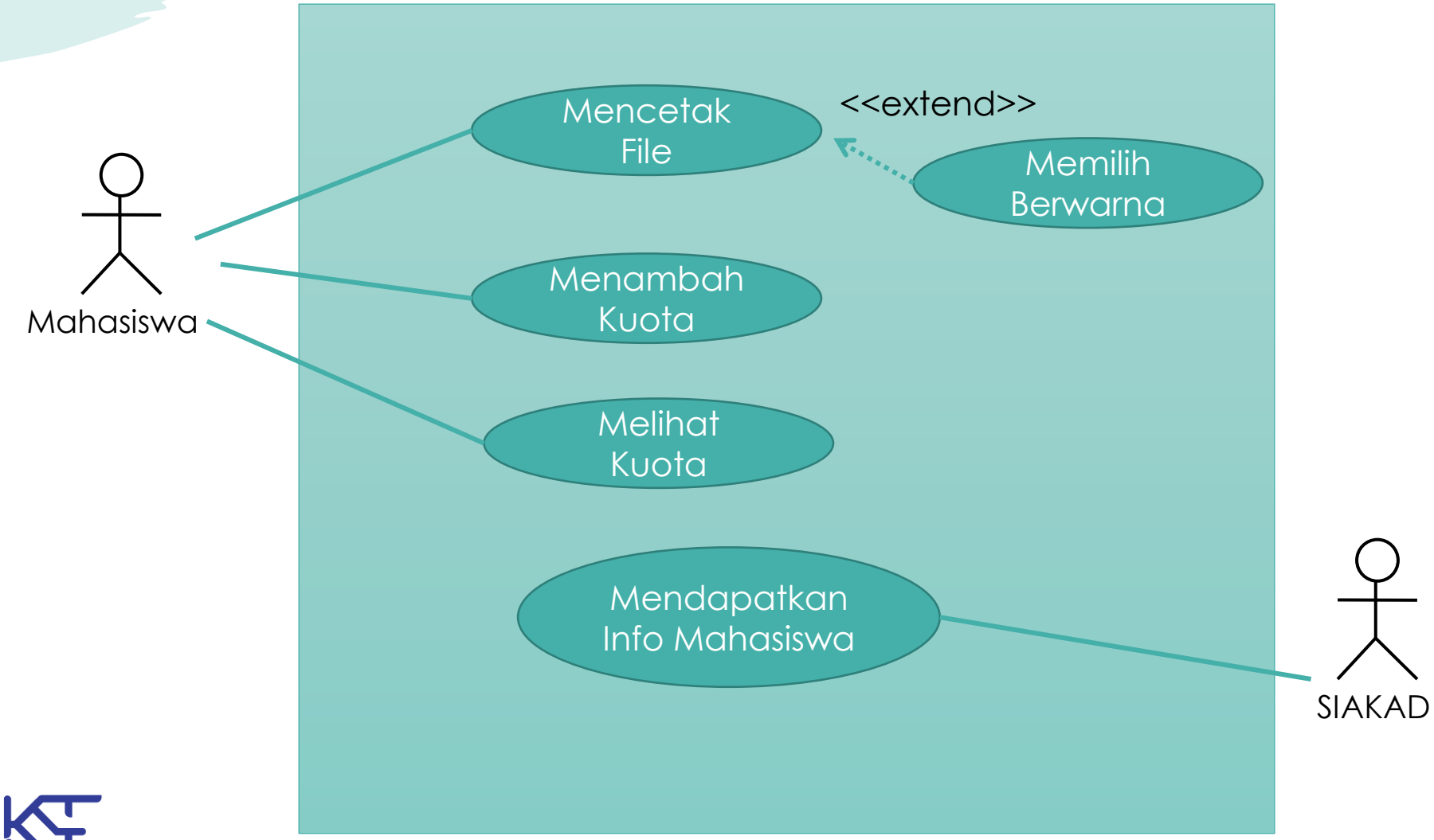
Gambarkan diagram use case

- Setiap ***use case*** harus **terhubung** dengan minimal **satu aktor**
- Jika *use case* terhubung dengan **lebih** dari satu aktor, maka harus diperjelas aktor yang men-***trigger*** *use case* pertama kali (gunakan **tanda panah**)
- Mungkin ada hubungan antar *use case*: ***include*** (*uses*) atau ***extends*** (*insert*)

Use Case untuk VM



Use Case Layanan Pencetakan Mahasiswa

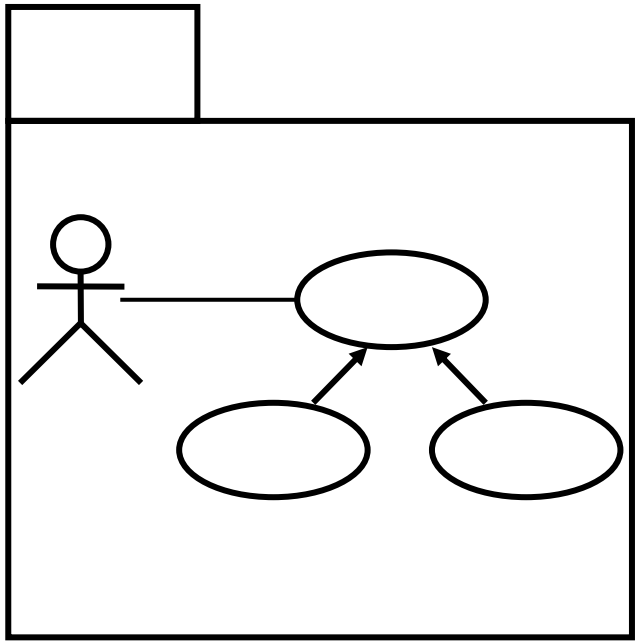


Membuat scenario

- Sekumpulan skenario dari pengguna digunakan untuk **menjelaskan pemakaian sistem**
- Setiap skenario **dilihat dari sudut pandang 'aktor'**
  - Aktor adalah orang atau sistem yang melakukan interaksi terhadap dengan software
- Setiap skenario **menjawab pertanyaan** berikut:
  - Siapa aktor utama, siapa aktor pendukung
  - Apa tujuan si aktor
  - Kondisi awal apa yang harus ada sebelum suatu cerita atau '*story*' dimulai
  - Apa tugas/fungsi utama yang dilakukan oleh si aktor
  - Apa tugas/fungsi tambahan yang dapat diberikan
  - Apa variasi yang memungkinkan dalam interaksi si aktor
  - Informasi dari sistem apa yang dibutuhkan, diproduksi atau diubah dari/oleh si aktor
  - Apakah si aktor harus memberitahukan sistem bila terjadi perubahan pada lingkungan eksternal?
  - Informasi apa yang diinginkan oleh si aktor dari sistem
  - Apakah si aktor ingin diberitahu bila ada perubahan yang di luar rencana?

Paket (Packages) dalam model Use-case

- **Paket** digunakan untuk **mengelompokkan** elemen-elemen yang terkait secara **semantik**
- Kegunaan:
  - *Use-case* lebih terstruktur, bisa disesuaikan dengan fungsi atau tipe pengguna
  - Sebagai **batasan** lingkup dari satu atau beberapa *use-case*
  - Paket dalam *use-case* juga bisa digunakan untuk
    - Menunjukkan urutan sistem
    - Konfigurasi sistem
    - *Delivery unit*
  - Pengalokasian sumber daya/kompetensi dalam tim pengembang dapat dibagi menjadi beberapa kelompok yang berbeda pada tempat yang berbeda



Contoh: Mesin Recycle

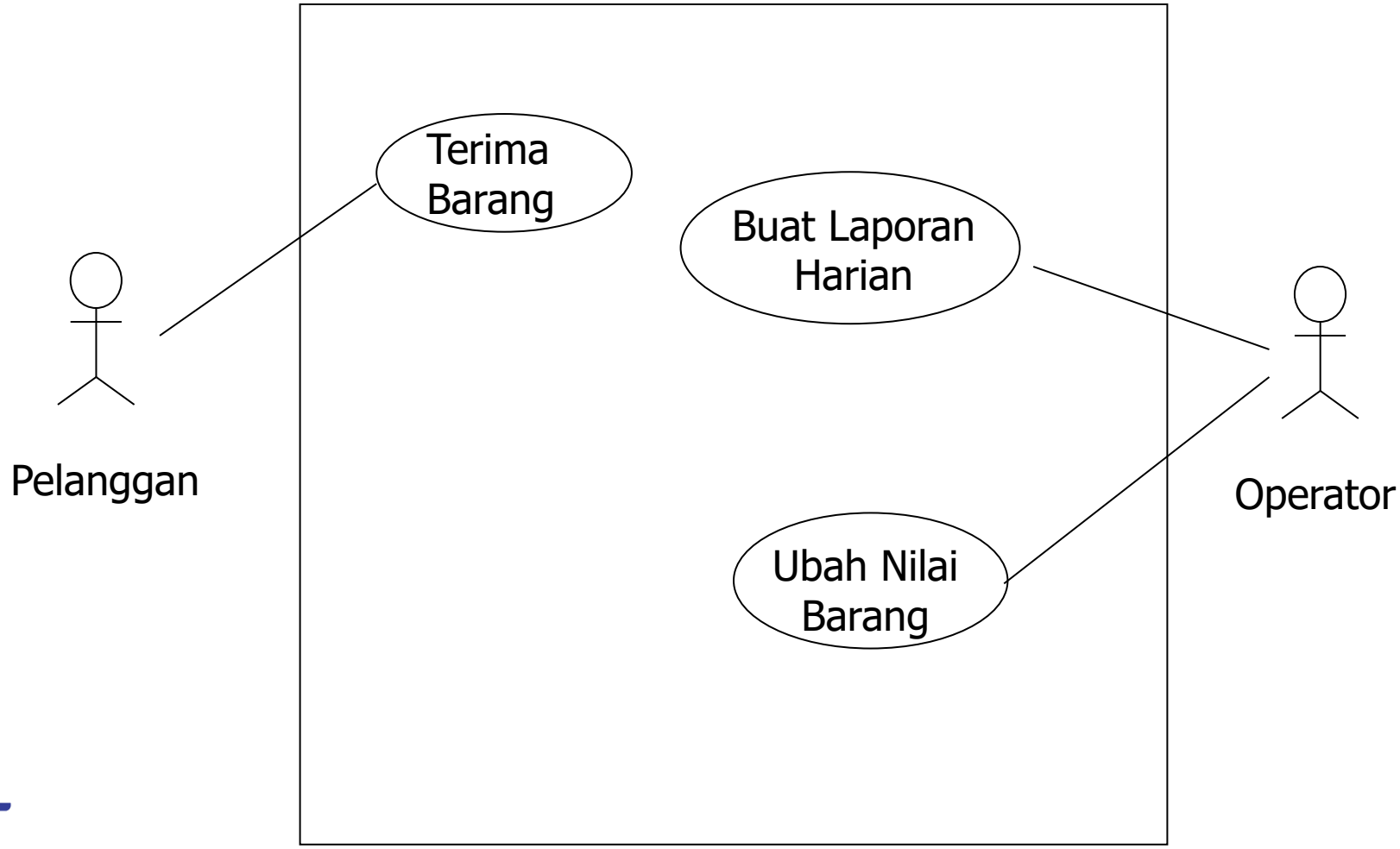
Sistem Mesin Recycle

- Suatu mesin melakukan *recycle* terhadap botol dan kaleng
- Mesin dapat digunakan oleh pengguna yang berbeda-beda
- Sistem akan mencatat tipe dan jumlah item yang dimasukkan
- Sistem akan mencetak tanda terima agar dapat diganti dengan uang oleh pengguna
- Operator dapat meminta laporan harian
- Operator dapat merubah nilai dari item (benda) yang dikembalikan
- (Operator akan diinformasikan bila ditemukan *malfunction*)

Mesin Recycle

- Aktor?
  - Pelanggan
    - Memasukkan kaleng, botol ke mesin *recycling*
    - Tekan tombol
    - Menerima tanda terima yang bisa ditukar dengan uang
  - Operator
- Ada berapa *use-case*?
  - Aksinya:
    - Memasukkan item (barang)
    - Tekan tombol
    - Mencetak tanda terima
  - Tapi dalam kenyataannya,
    - Ada tiga kejadian, tapi masing-masing saling terkait, dan bagi pelanggan detil kejadian itu tidak terlalu penting.
    - Agar pelanggan mudah mengerti, maka ketiga aksi itu dapat dilihat sebagai proses yang lengkap
      - Mulai dari memasukkan barang(item), tekan tombol hingga dapat tanda terima
  - Jadi sebagai suatu instansiasi yang lengkap hanya akan ada satu use-case
- Gambarkan diagram *use case* (slide berikutnya)
- Detilkan skenarionya

Contoh: Sistem Mesin Recycle



Contoh: Jalur Dasar untuk use-case "Terima Barang"

- Kondisi Awal(*Pre-condition/Initial State*): Pelanggan mau meletakkan botol dan kaleng ke mesin
  - Pelanggan meletakkan setiap barang ke mesin
  - Sistem akan menaikkan jumlah barang yang diterima, juga mencatat jumlah total barang secara keseluruhan
  - Pelanggan menekan tombol selesai untuk mendapatkan tanda terima
  - Sistem akan mencetak tanda terima berisi jumlah barang yang dimasukkan.
- Jalur alternatif
  - Pada langkah pertama, barang mungkin terjepit di slot mesin
    - Pada kasus ini, pelanggan harus diinformasikan, operator juga diberikan notifikasi, dan mesin akan mencetak tanda terima barang yang sudah terlebih dulu dimasukkan
- Kondisi Akhir (*Post-condition/final state*): *Use-case* selesai setelah tombol tanda terima (tombol selesai) ditekan

KNOWLEDGE & SOFTWARE ENGINEERING

Jalur dasar: Buat Laporan harian

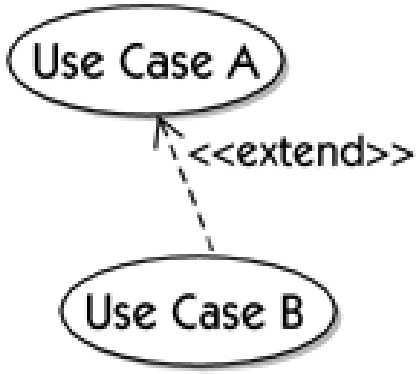
- Kondisi-awal: Operator mau mencetak laporan harian untuk semua barang yang dimasukkan
  - Sistem akan mencetak jumlah barang yang diterima untuk setiap tipe
  - Sistem akan mencetak jumlah barang
  - Sistem akan di-*reset* kembali nol
- Kondisi-akhir: Jumlah harian di-set nol setelah *use-case* selesai

Jalur Dasar: Ubah Nilai Barang

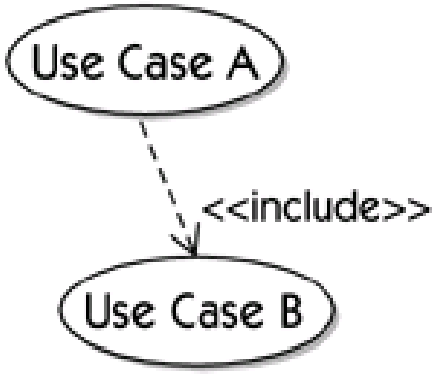
- Kondisi-awal: Operator mau mengubah nilai keterangan barang
  - Nilai jumlah setiap barang yang akan diubah
  - Ukuran barang yang akan dirubah
  - Tipe barang baru dapat ditambahkan
- Kondisi-akhir: Nilai keterangan barang baru berubah

Ketergantungan antar use-case

- Hubungan ***extend*** mendefinisikan suatu instansiasi dari suatu *use-case* yang menunjukkan adanya perilaku (*behaviour*) **tambahan**
- Hubungan ***include*** mendefinisikan hubungan **langsung** dua *use-case*
  - *use-case* yang dimasukkan akan dilakukan oleh *use-case* dasarnya



use-case B **mungkin** dilakukan oleh *use-case* A

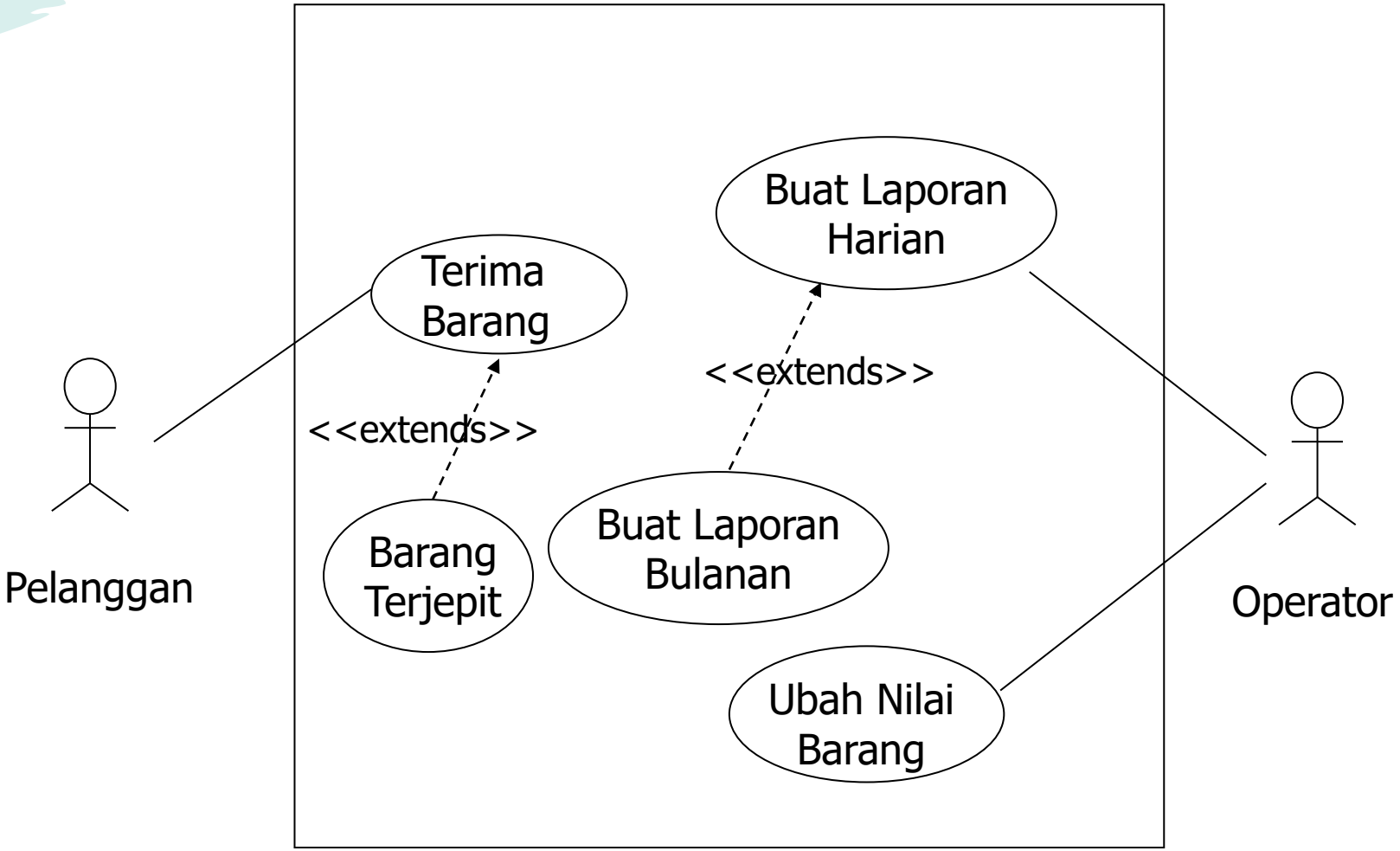


use-case B **dilibatkan** ketika *use-case* A dilakukan

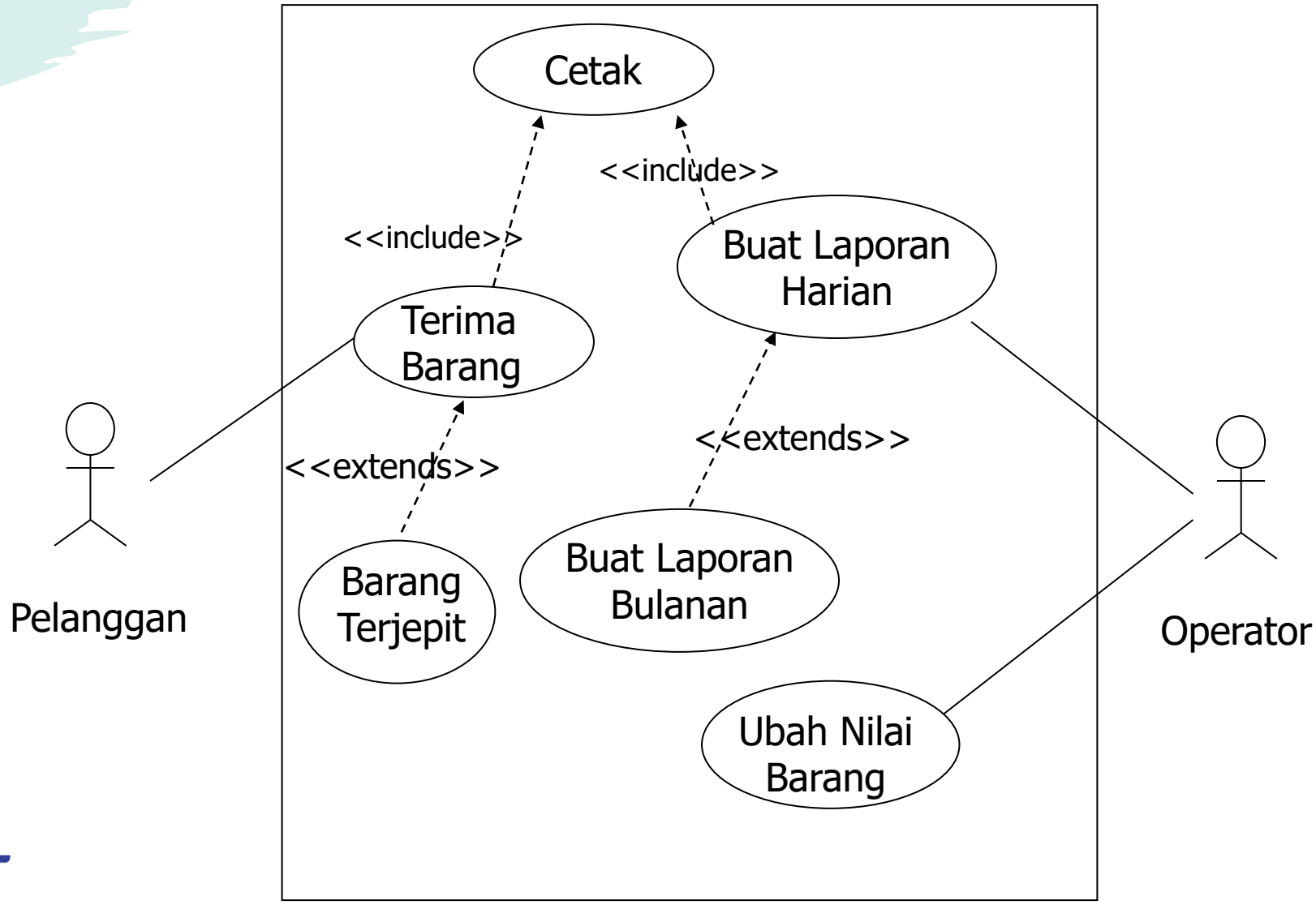
Extend/Include

- Dalam model *use-case*, dimungkin adanya pilihan *event* alternatif ataupun ada *use-case* yang harus dilakukan jika suatu *use-case* dieksekusi
  - **Alternatif** *use-case* ditunjukkan sebagai **<<extends>>**
    - Extend ini menjadi bagian yang **optional** (bisa dilakukan dan bisa tidak)
  - *Use-case* yang **harus** dilakukan ditunjukkan sebagai **<<include>>**
    - **Harus dilakukan**
- Walau jarang terjadi, tetapi *use-case extends* dan *include* bisa terjadi.
  - Digunakan untuk kasus tertentu
- Kenapa ada *extends/include*?
  - Karena perkembangan kebutuhan, sehingga harus ditambahkan fitur/ fungsionalitas baru
  - *use-case* perlu dikembangkan untuk mendapatkan aksi tambahan dalam suatu kondisi lain

Contoh dengan Extends

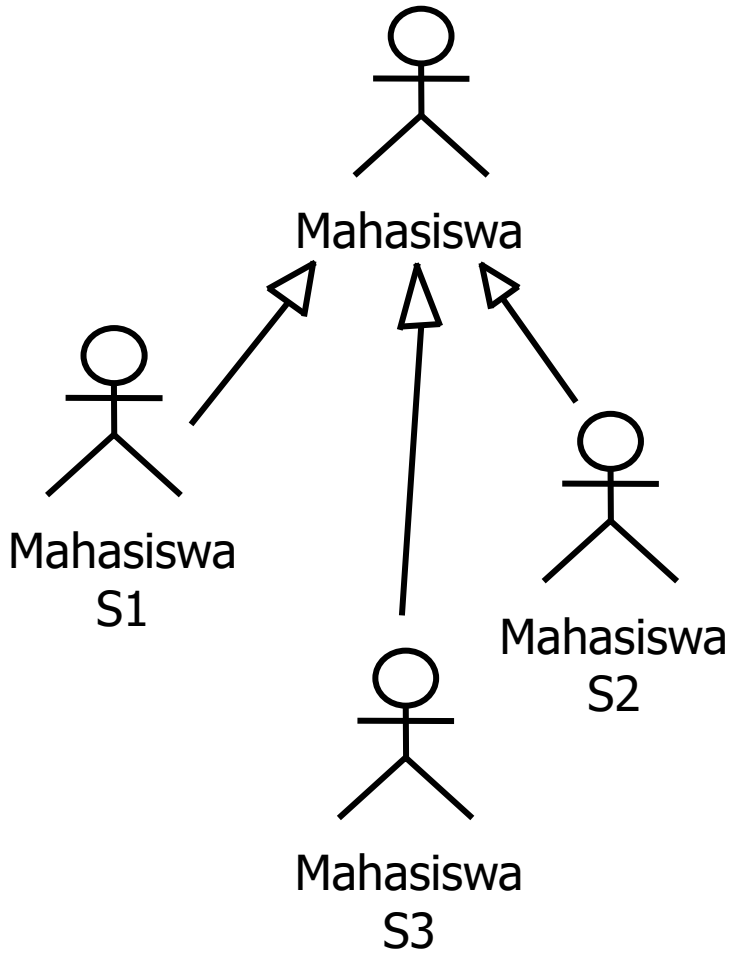


Contoh dengan Include

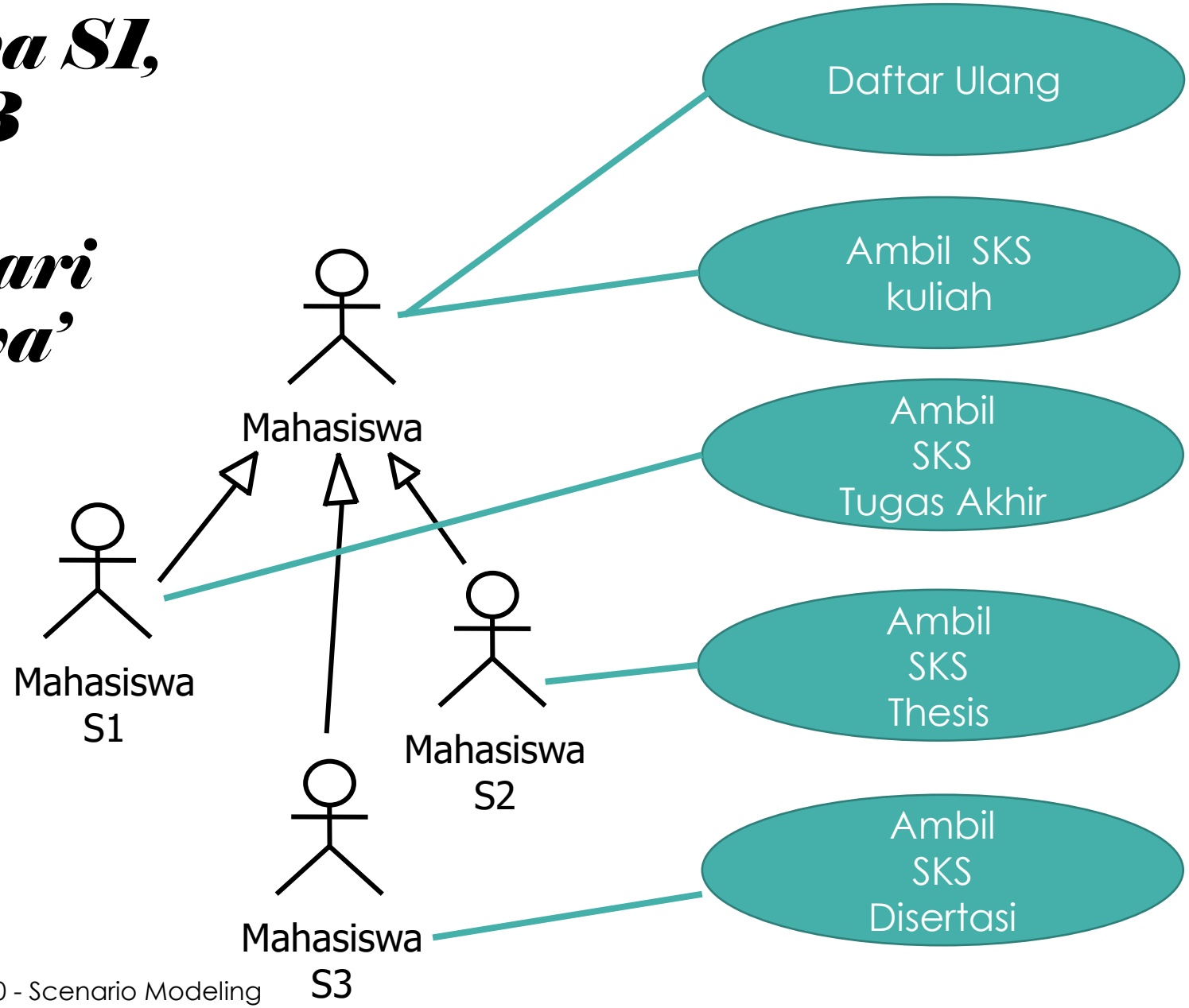


Generalisasi Aktor (Actor Generalization/Actor Inheritance)

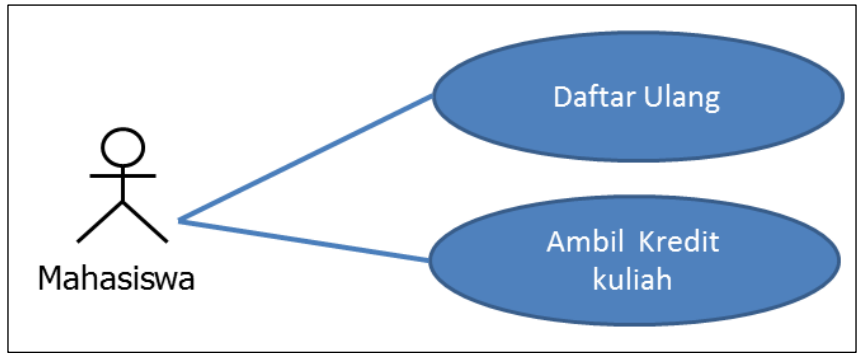
- Beberapa aktor dapat memiliki **peran** yang **sama** pada suatu *use-case*
- Contoh:
  - Ada mahasiswa S1, S2 dan S3 yang ketiganya terdaftar suatu kuliah
  - Ketiga akan terlihat sebagai entitas eksternal oleh *use-case* 'Daftar Ulang' atau 'Ambil mata kuliah'
  - Mahasiswa S1, S2 atau S3 dapat dimodelkan sebagai 'Mahasiswa' saja, karena ketiganya memiliki banyak kesamaan
- Hal ini disebut '**generalisasi aktor**'

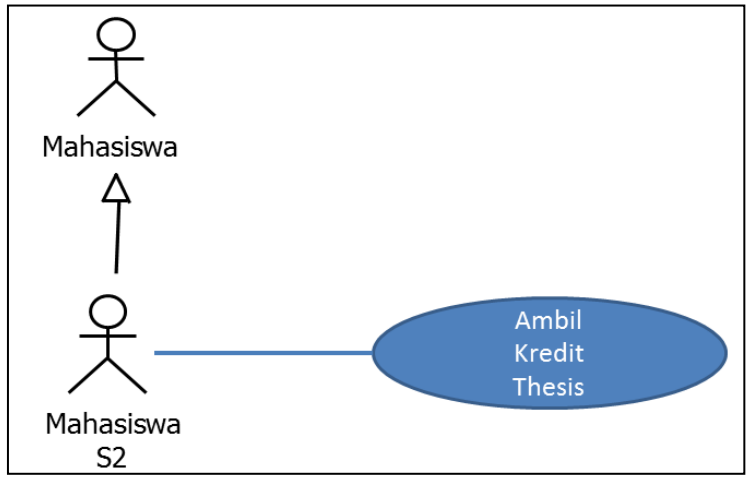


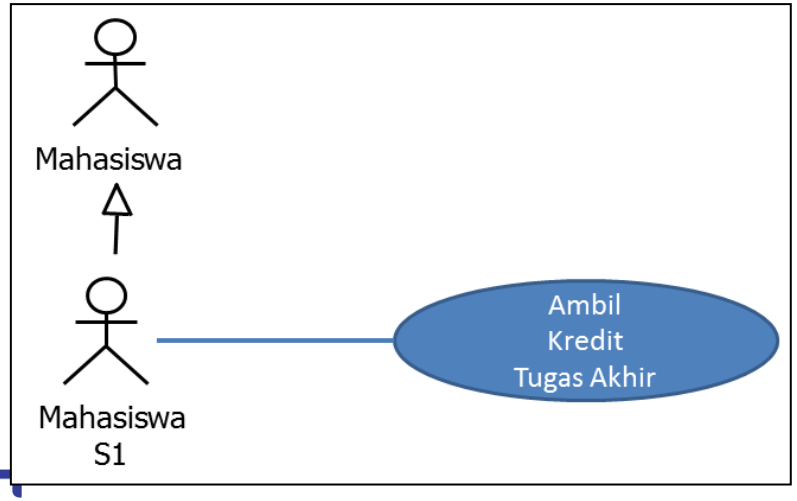
Mahasiswa S1, S2, dan S3 adalah turunan dari 'Mahasiswa'

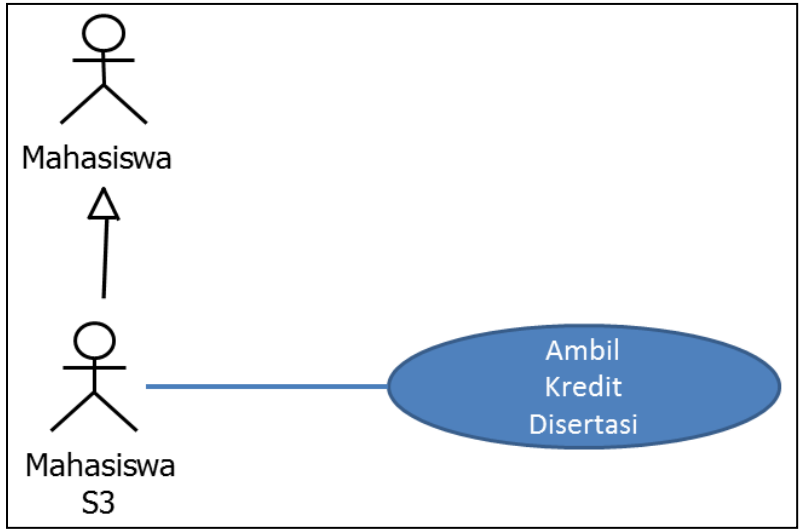


Generalisasi Aktor dapat dipisahkan penggambarannya





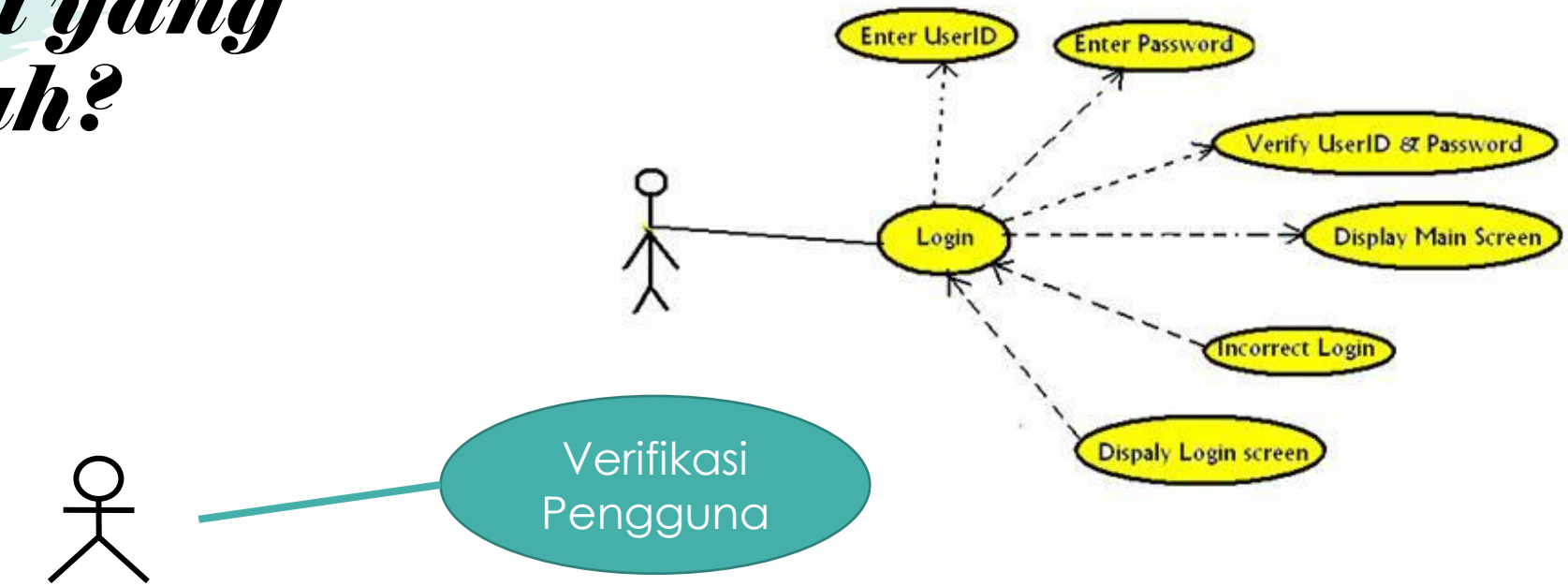




Pemakaian Use-case

- Dalam pengembangan perangkat lunak, *Use-case* akan berperan:
  - Membantu proses iterasi untuk mengerti masalah
  - Membantu mencari prioritas pengembangan

Apa yang salah?



Skenario:

Display Login Screen
Enter User Id
Enter Password
Verify User Id dan Password
If LoginError then Incorrect Login
Else display Main Screen

Latihan membuat diagram use-case

Akan dibangun sebuah perangkat lunak untuk mendukung proses pendaftaran ulang mahasiswa secara online. Melalui aplikasi tersebut, mahasiswa dapat mengajukan usulan pengambilan matakuliah.

Selanjutnya, dosen wali dapat melihat usulan pengambilan matakuliah untuk disetujui/ditolak. Usulan yang ditolak dapat direvisi kembali oleh mahasiswa.

Usulan yang telah disetujui wali dapat langsung diproses oleh Petugas Administrasi untuk pencetakan KSM. KSM hanya bisa dicetak apabila status pembayaran SPP mahasiswa sudah beres. Informasi status pembayaran SPP diperoleh dari perangkat lunak lain yaitu SISKEU (Sistem Informasi Keuangan). Perangkat lunak ini juga akan berhubungan dengan perangkat lunak SIKAD (Sistem Informasi Akademik) untuk mendapatkan informasi tentang matakuliah yang ditawarkan pada semester tersebut, serta informasi transkrip nilai mahasiswa, agar dosen wali mendapatkan referensi untuk menyetujui/menolak usulan pengambilan matakuliah.

Cari Aktor!

Akan dibangun sebuah perangkat lunak untuk mendukung proses pendaftaran ulang mahasiswa secara online. Melalui aplikasi tersebut, mahasiswa dapat mengajukan usulan pengambilan matakuliah.

Selanjutnya, dosen wali dapat melihat usulan pengambilan matakuliah untuk disetujui/ditolak. Usulan yang ditolak dapat direvisi kembali oleh mahasiswa.

Usulan yang telah disetujui wali dapat langsung diproses oleh Petugas Administrasi untuk pencetakan KSM. KSM hanya bisa dicetak apabila status pembayaran SPP mahasiswa sudah beres. Informasi status pembayaran SPP diperoleh dari perangkat lunak lain yaitu SISKEU (Sistem Informasi Keuangan). Perangkat lunak ini juga akan berhubungan dengan perangkat lunak SIKAD (Sistem Informasi Akademik) untuk mendapatkan informasi tentang matakuliah yang ditawarkan pada semester tersebut, serta informasi transkrip nilai mahasiswa, agar dosen wali mendapatkan referensi untuk menyetujui/menolak usulan pengambilan matakuliah.

Aktor

Akan dibangun sebuah perangkat lunak untuk mendukung proses pendaftaran ulang **mahasiswa** secara online. Melalui aplikasi tersebut, mahasiswa dapat mengajukan usulan pengambilan matakuliah.

Selanjutnya, **dosen wali** dapat melihat usulan pengambilan matakuliah untuk disetujui/ditolak. Usulan yang ditolak dapat direvisi kembali oleh mahasiswa.

Usulan yang telah disetujui wali dapat langsung diproses oleh **Petugas Administrasi** untuk pencetakan KSM. KSM hanya bisa dicetak apabila status pembayaran SPP mahasiswa sudah beres. Informasi status pembayaran SPP diperoleh dari perangkat lunak lain yaitu **SISKEU** (Sistem Informasi Keuangan). Perangkat lunak ini juga akan berhubungan dengan perangkat lunak **SIKAD** (Sistem Informasi Akademik) untuk mendapatkan informasi tentang matakuliah yang ditawarkan pada semester tersebut, serta informasi transkrip nilai mahasiswa, agar dosen wali mendapatkan referensi untuk menyetujui/menolak usulan pengambilan matakuliah.

Aktor

- Mahasiswa
- Dosen Wali
- Petugas Administrasi
- SISKEU
- SIKAD

Cari Use-case!

Akan dibangun sebuah perangkat lunak untuk mendukung proses pendaftaran ulang mahasiswa secara online. Melalui aplikasi tersebut, mahasiswa dapat mengajukan usulan pengambilan matakuliah.

Selanjutnya, dosen wali dapat melihat usulan pengambilan matakuliah untuk disetujui/ditolak. Usulan yang ditolak dapat direvisi kembali oleh mahasiswa.

Usulan yang telah disetujui wali dapat langsung diproses oleh Petugas Administrasi untuk pencetakan KSM. KSM hanya bisa dicetak apabila status pembayaran SPP mahasiswa sudah beres. Informasi status pembayaran SPP diperoleh dari perangkat lunak lain yaitu SISKEU (Sistem Informasi Keuangan). Perangkat lunak ini juga akan berhubungan dengan perangkat lunak SIKAD (Sistem Informasi Akademik) untuk mendapatkan informasi tentang matakuliah yang ditawarkan pada semester tersebut, serta informasi transkrip nilai mahasiswa, agar dosen wali mendapatkan referensi untuk menyetujui/menolak usulan pengambilan matakuliah.

Use-case

Akan dibangun sebuah perangkat lunak untuk mendukung proses pendaftaran ulang mahasiswa secara online. Melalui aplikasi tersebut, mahasiswa dapat **mengajukan usulan** pengambilan matakuliah.

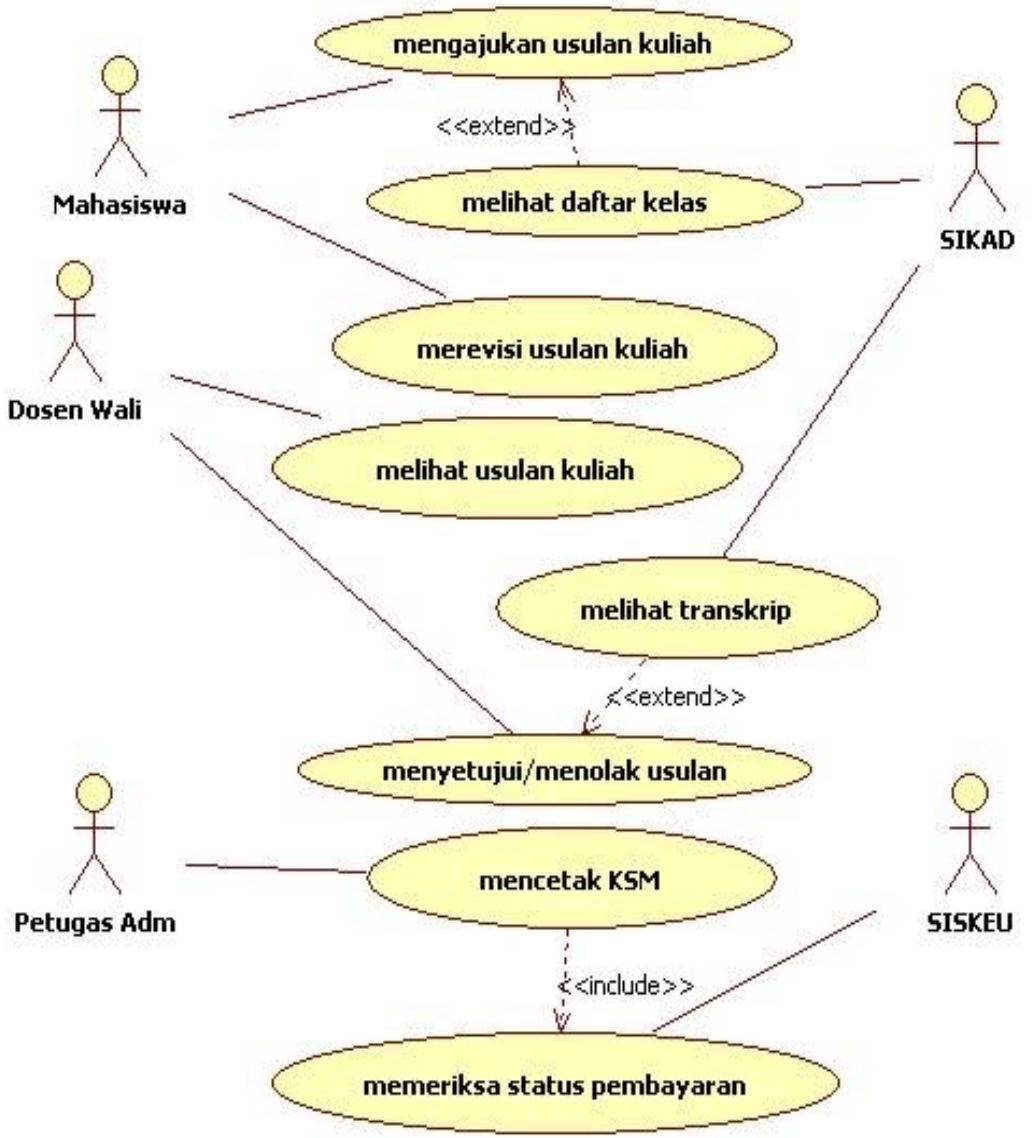
Selanjutnya, dosen wali dapat **melihat usulan** pengambilan matakuliah untuk **disetujui/ditolak**. Usulan yang ditolak dapat **direvisi** kembali oleh mahasiswa.

Usulan yang telah disetujui wali dapat langsung diproses oleh Petugas Administrasi untuk **pencetakan** KSM. KSM hanya bisa dicetak apabila **status pembayaran** SPP mahasiswa sudah beres. Informasi status pembayaran SPP diperoleh dari perangkat lunak lain yaitu SISKEU (Sistem Informasi Keuangan). Perangkat lunak ini juga akan berhubungan dengan perangkat lunak SIKAD (Sistem Informasi Akademik) untuk **mendapatkan informasi tentang matakuliah** yang ditawarkan pada semester tersebut, serta **informasi transkrip** nilai mahasiswa, agar dosen wali mendapatkan referensi untuk menyetujui/menolak usulan pengambilan matakuliah.

Use-case

- Mengajukan Usulan
- Melihat Usulan
- Menyetujui Usulan
- Menolak Usulan
- Merevisi Usulan
- Memeriksa Status Pembayaran
- Melihat Daftar Kelas
- Melihat Transkrip
- Mencetak KSM

Use-case Diagram



Skenario Mengajukan Usulan

- Mahasiswa memilih menu entri usulan kuliah
- Sistem menampilkan form entri FRS
- Mahasiswa mengisikan kode kuliah
- Sistem menampilkan informasi detil matakuliah (nama, sks)
- Mahasiswa menekan tombol SIMPAN
- Sistem menyimpan data usulan ke dalam basisdata

Alternatif skenario Mengajukan Usulan dan Melihat Daftar Kelas

- Mahasiswa memilih menu entri usulan kuliah
- Mahasiswa memilih untuk melihat daftar kelas
- Sistem menampilkan daftar kelas yang dibuka
- Mahasiswa memilih matakuliah dari daftar
- Mahasiswa menekan tombol SIMPAN
- Sistem menyimpan data usulan ke dalam basisdata

Alternatif skenario... (2)

- Mahasiswa memilih menu entri usulan kuliah
- Sistem menampilkan form entri FRS
- Mahasiswa mengisikan kode kuliah
- Sistem menampilkan pesan bahwa kelas untuk kuliah tersebut tidak dibuka

Deskripsi Use Case

Use Case: Mengajukan Usulan Kuliah
Iteration : ke-2,
Modifikasi terakhir 1 Maret 2018
Primary Actor: Mahasiswa
Goal in Context: Untuk pengajuan usulan kuliah oleh mahasiswa
Preconditions: Mahasiswa sudah terdaftar dan mahasiswa sudah memasukkan nama user dan password sebelumnya
Trigger: Jika Mahasiswa memutuskan untuk mengambil kuliah di awal semester

Scenario:

1. Mahasiswa memilih menu entri usulan kuliah
2. Sistem menampilkan form entri FRS
3. Mahasiswa mengisikan kode kuliah
4. Sistem menampilkan informasi detil matakuliah (nama, sks)
5. Mahasiswa menekan tombol SIMPAN
6. Sistem menyimpan data usulan ke dalam basisdata

Exception:

1. Mahasiswa memilih menu entri usulan kuliah
2. Mahasiswa memilih untuk melihat daftar kelas
3. Sistem menampilkan daftar kelas yang dibuka
4. Mahasiswa memilih matakuliah dari daftar
5. Mahasiswa menekan tombol SIMPAN
6. Sistem menyimpan data usulan ke dalam basisdata

Priority: Prioritas sedang
When available: Iterasi ketiga
Secondary Actor: tidak ada
Open Issues:

1. Bagaimana mekanisme mendeteksi, jika ada suatu kuliah memiliki prerequisite kuliah lain
2. Bagaimana jika mahasiwa ingin membatalkan usulan

Activity Diagram

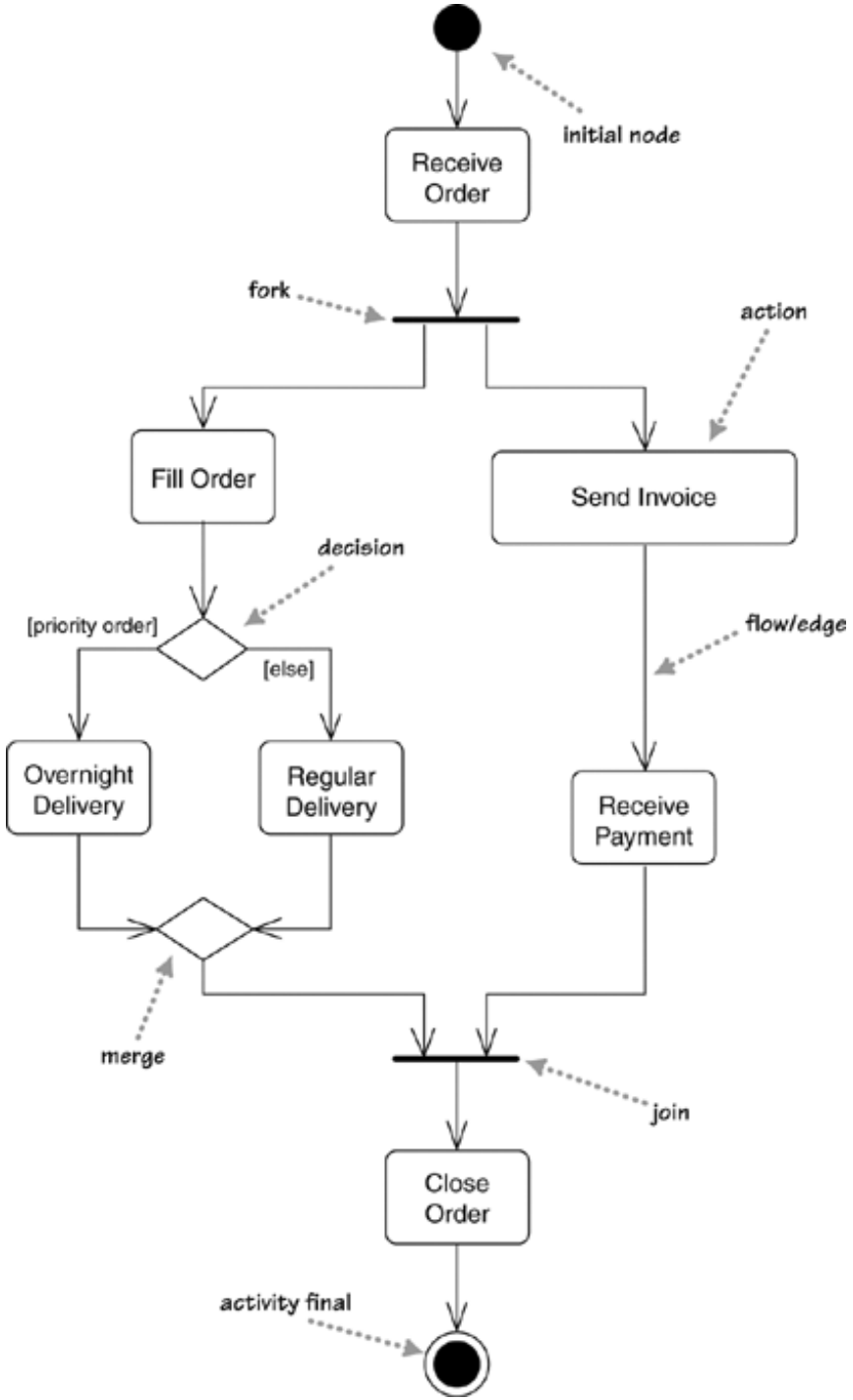
KNOWLEDGE & SOFTWARE ENGINEERING

Activity Diagram

- Diagram aktivitas menjelaskan
  - Urutan proses prosedural
  - Urutan bisnis proses
  - Urutan kerja (*work flow*)

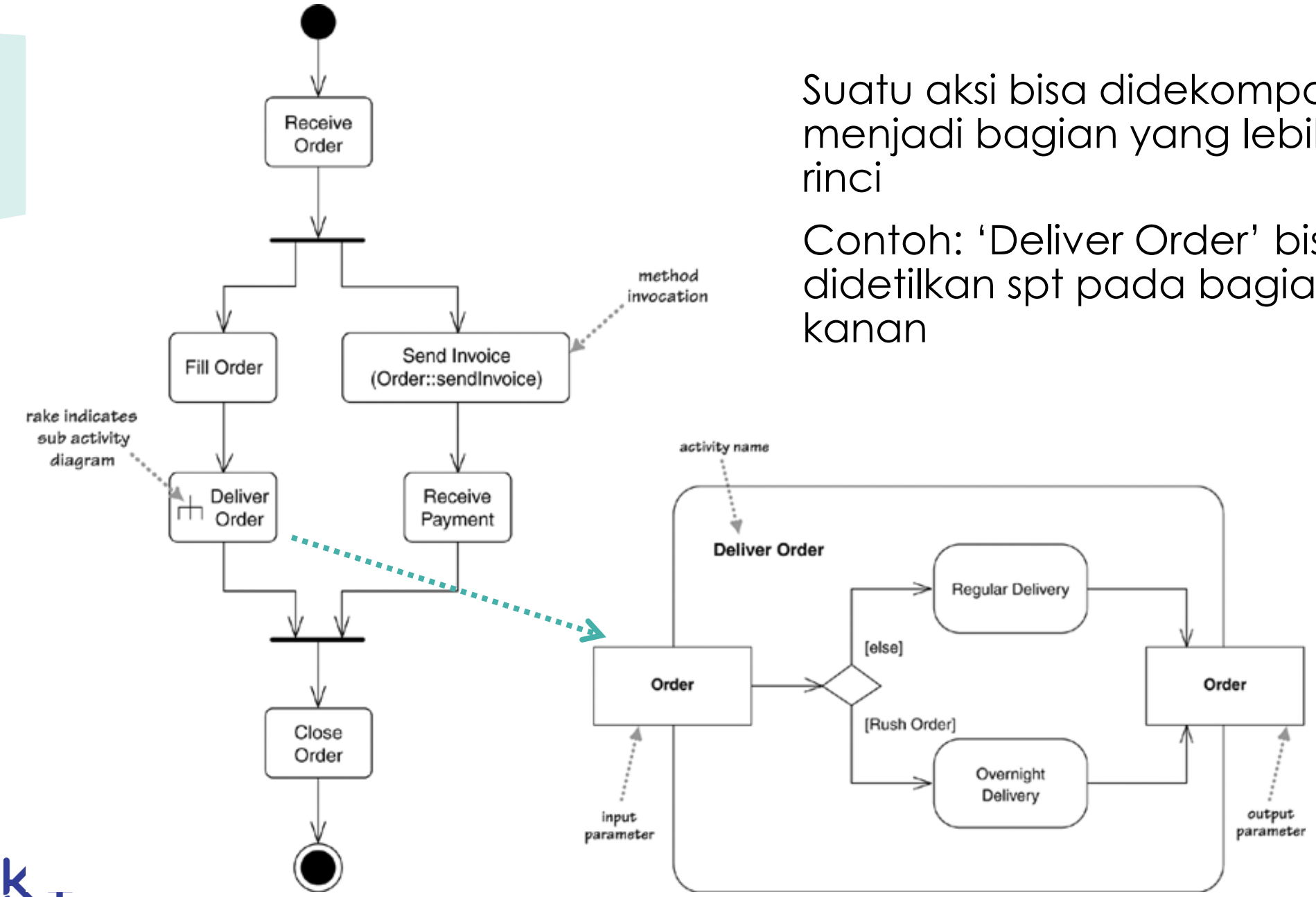
- Diagram aktivitas ini mirip seperti '*Flow chart*'
  - Tetapi *flow-chart* tidak mengenal 'perilaku paralel'/'konkuren'

Contoh


initial node
Receive Order
fork
action
Fill Order
Send Invoice
decision
[priority order]
[else]
flow/edge
Overnight Delivery
Regular Delivery
Receive Payment
merge
join
Close Order
activity final

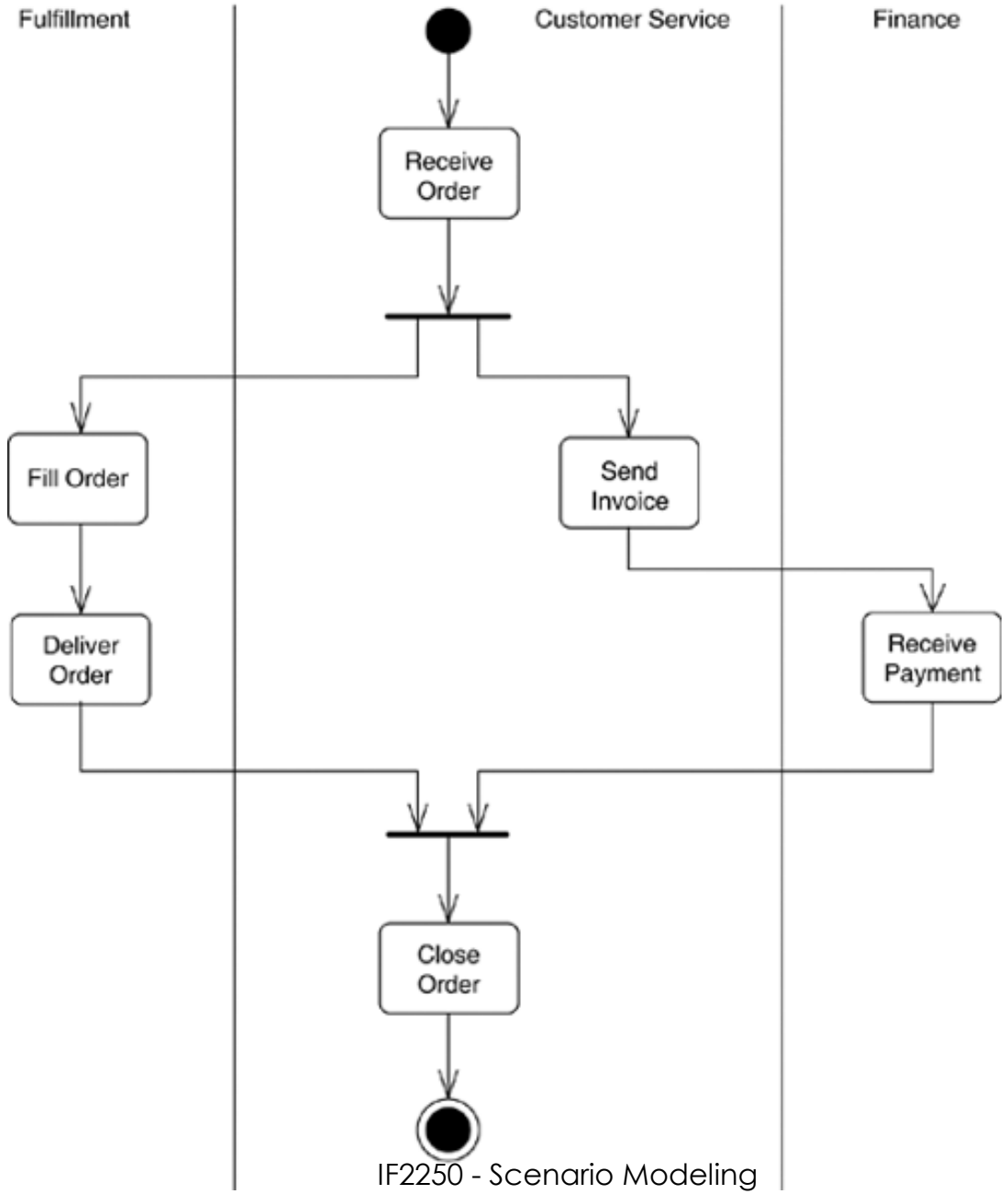
Suatu aksi bisa didekomposisi menjadi bagian yang lebih rinci

Contoh: 'Deliver Order' bisa didetilkan spt pada bagian kanan



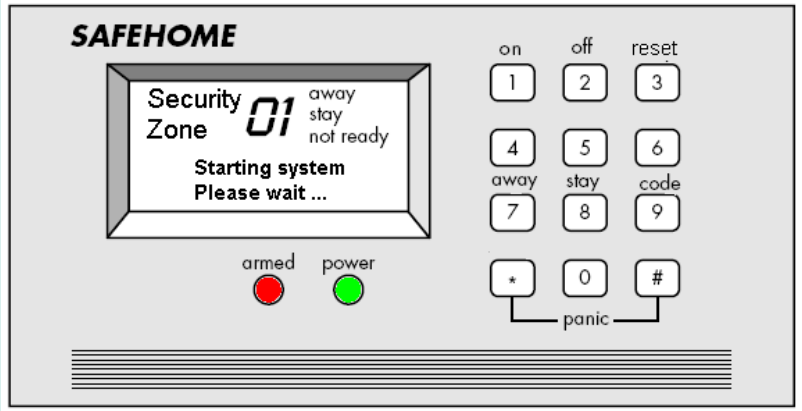
Partisi

Diagram aktivitas dapat dipartisi berdasarkan 'siapa' yang melakukan 'apa'



Contoh Pengembangan Use-Case untuk Safe Home

(Referensi: R. Pressman, "Software Engineering: A Practitioner's Approach" 7th edition)



See SEPA 193,231pg for more details

Skenario Awal (Safe Home)

The scene: A meeting room, during the second requirements gathering meeting.

The players: Jamie Lazar, software team member; Ed Robbins, software team member; Doug Miller, software engineering manager; three members of marketing; a product engineering representative; and a facilitator.

The conversation:

Facilitator: It's time that we begin talking about the SafeHome surveillance function. Let's develop a user scenario for access to the surveillance function.

Jamie: Who plays the role of the actor on this?

Facilitator: I think Meredith (a marketing person) has been working on that functionality. Why don't you play the role?

Meredith: You want to do it the same way we did it last time, right?

Facilitator: Right . . . same way.

Meredith: Well, obviously the reason for surveillance is to allow the homeowner to check out the house while he or she is away, to record and play back video that is captured . . . that sort of thing.

Ed: Will we use compression to store the video?

Facilitator: Good question, Ed, but let's postpone implementation issues for now. Meredith?

Meredith: Okay, so basically there are two parts to the surveillance function . . . the first configures the system including laying out a floor plan—we have to have tools to help the homeowner do this—and the second part is the actual surveillance function itself. Since the layout is part of the configuration activity, I'll focus on the surveillance function.

Facilitator (smiling): Took the words right out of my mouth.

Meredith: Um . . . I want to gain access to the surveillance function either via the PC or via the Internet. My feeling is that the Internet access would be more frequently used. Anyway, I want to be able to display camera views on a PC and control pan and zoom for a specific camera. I specify the camera by selecting it from the house floor plan. I want to selectively record camera output and replay camera output. I also want to be able to block access to one or more cameras with a specific password. I also want the option of seeing small windows that show views from all cameras and then be able to pick the one I want enlarged.

Jamie: Those are called thumbnail views.

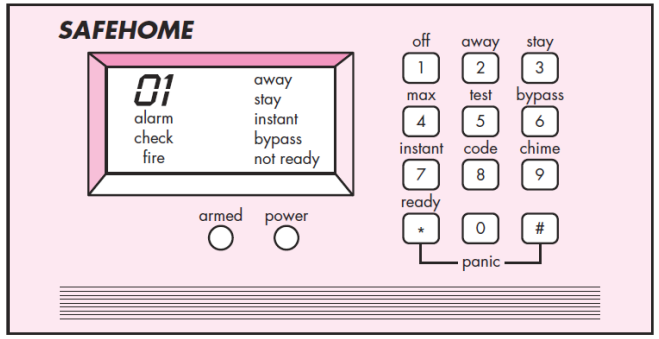
Meredith: Okay, then I want thumbnail views of all the cameras. I also want the interface for the surveillance function to have the same look and feel as all other SafeHome interfaces. I want it to be intuitive, meaning I don't want to have to read a manual to use it.

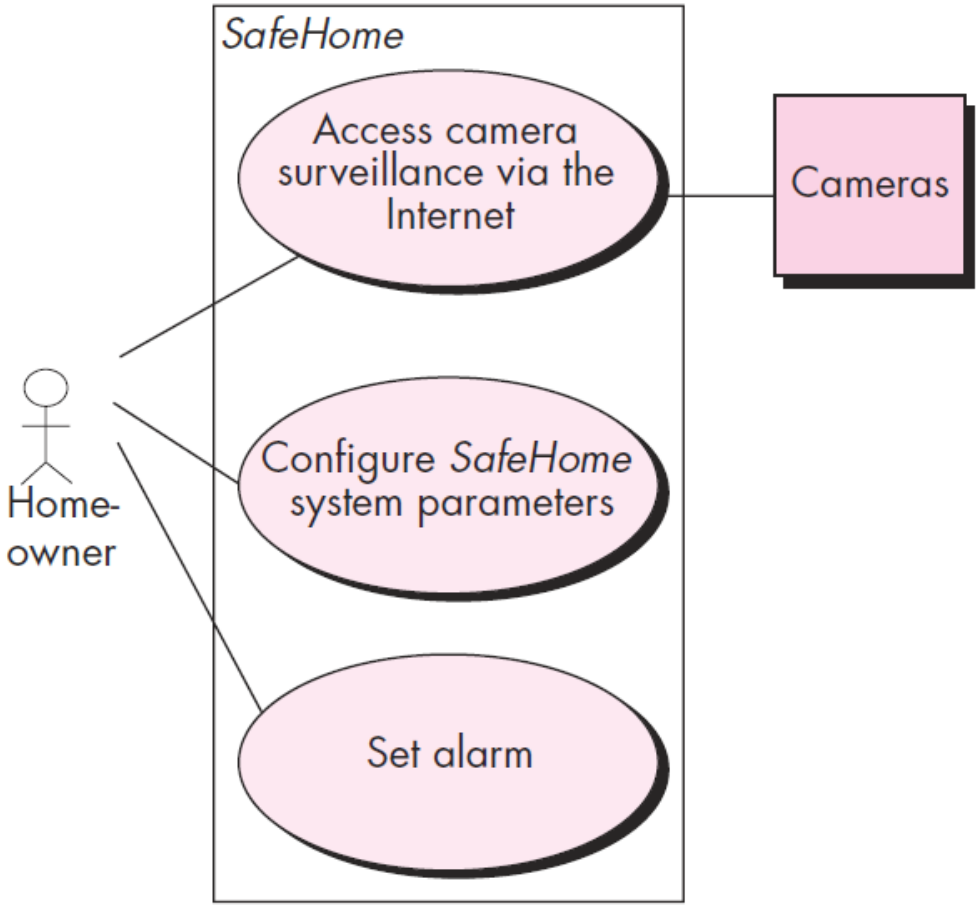
Facilitator: Good job. Now, let's go into this function in a bit more detail . . .

Fungsi "HomeOwner"

- Select camera to view.
- Request thumbnails from all cameras.
- Display camera views in a PC window.
- Control pan and zoom for a specific camera.
- Selectively record camera output.
- Replay camera output.
- Access camera surveillance via the Internet.

Diagram Use-case Safe Home





Skenario use case

SafeHome

Use Case Template for Surveillance

Use case: Access camera surveillance via the Internet—display camera views (ACS-DCV)

Iteration: 2, last modification: January 14 by V. Raman.

Primary actor: Homeowner.

Goal in context: To view output of camera placed throughout the house from any remote location via the Internet.

Preconditions: System must be fully configured; appropriate user ID and passwords must be obtained.

Trigger: The homeowner decides to take a look inside the house while away.

Scenario:

1. The homeowner logs onto the *SafeHome Products* website.
2. The homeowner enters his or her user ID.
3. The homeowner enters two passwords (each at least eight characters in length).
4. The system displays all major function buttons.
5. The homeowner selects the "surveillance" from the major function buttons.
6. The homeowner selects "pick a camera."
7. The system displays the floor plan of the house.
8. The homeowner selects a camera icon from the floor plan.
9. The homeowner selects the "view" button.
10. The system displays a viewing window that is identified by the camera ID.
11. The system displays video output within the viewing window at one frame per second.

Exceptions:

1. ID or passwords are incorrect or not recognized—see use case **Validate ID and passwords.**
2. Surveillance function not configured for this system—system displays appropriate error message; see use case **Configure surveillance function.**
3. Homeowner selects "View thumbnail snapshots for all camera"—see use case **View thumbnail snapshots for all cameras.**
4. A floor plan is not available or has not been configured—display appropriate error message and see use case **Configure floor plan.**
5. An alarm condition is encountered—see use case **Alarm condition encountered.**

Priority: Moderate priority, to be implemented after basic functions.

When available: Third increment.

Frequency of use: Moderate frequency.

Channel to actor: Via PC-based browser and Internet connection.

Secondary actors: System administrator, cameras.

Channels to secondary actors:

1. System administrator: PC-based system.
2. Cameras: wireless connectivity.

Open issues:

1. What mechanisms protect unauthorized use of this capability by employees of *SafeHome Products*?
2. Is security sufficient? Hacking into this feature would represent a major invasion of privacy.
3. Will system response via the Internet be acceptable given the bandwidth required for camera views?
4. Will we develop a capability to provide video at a higher frames-per-second rate when high-bandwidth connections are available?

Diagram Aktivitas untuk Use case Access camera surveillance via the internet

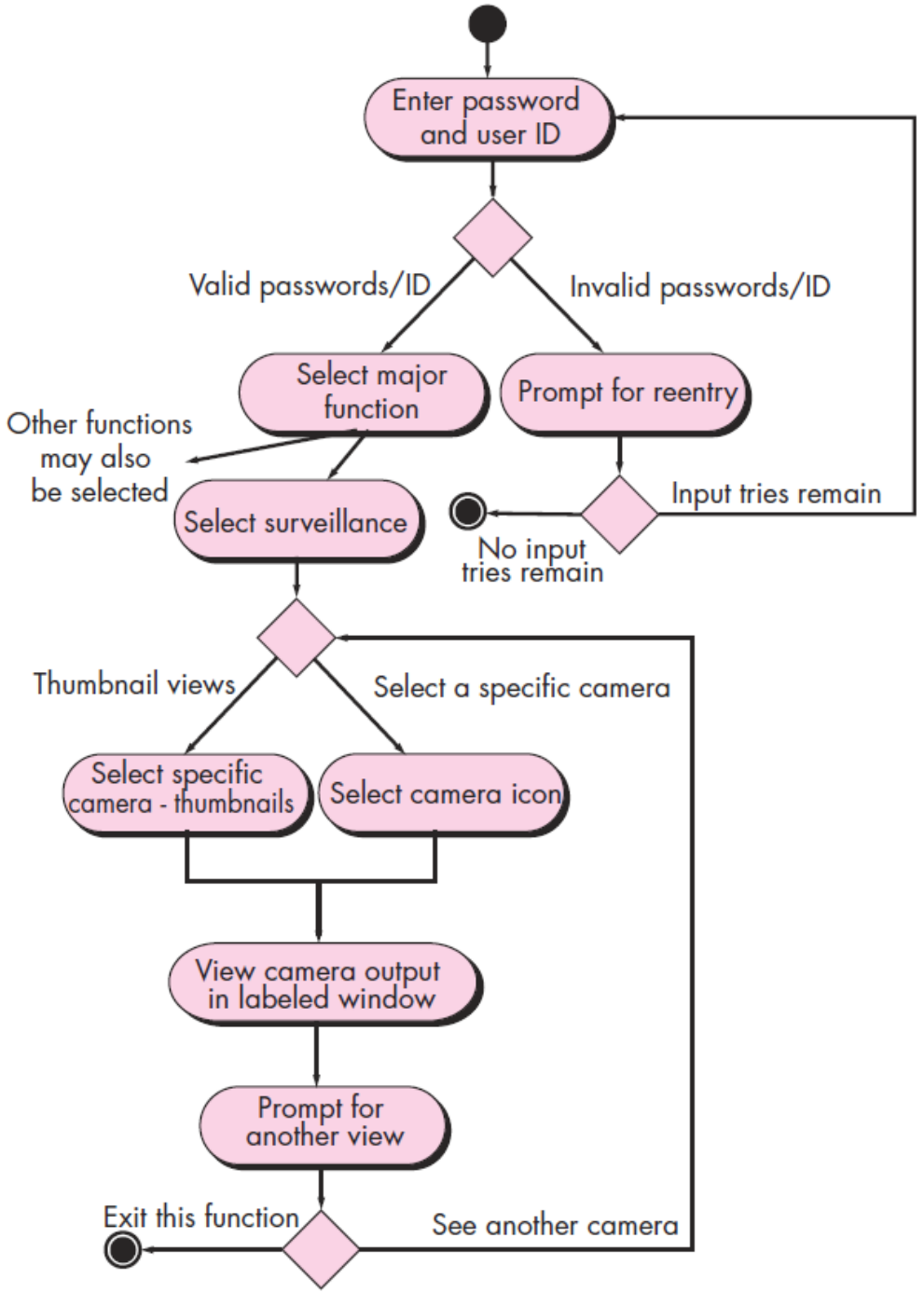
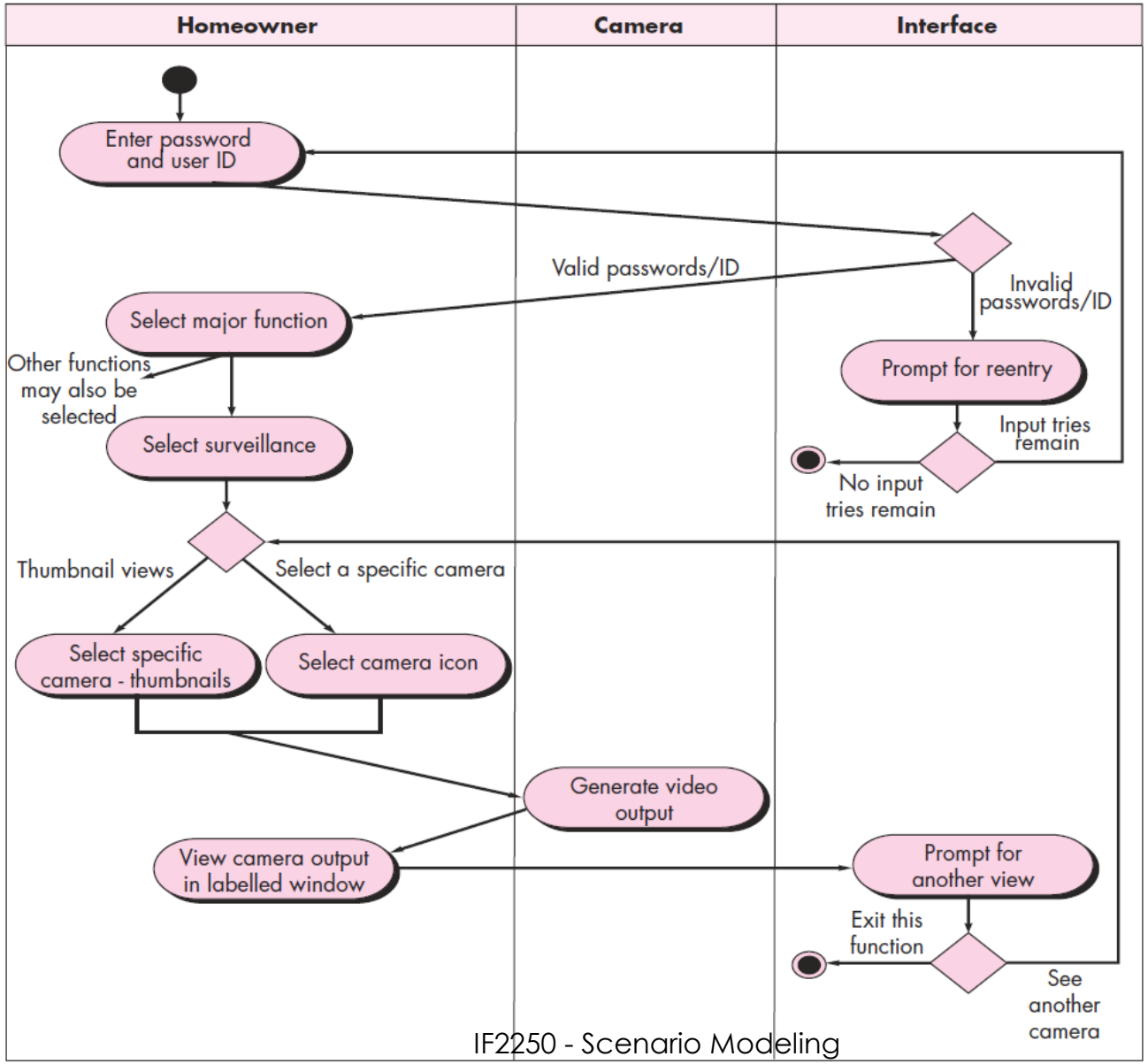


Diagram Aktivitas dalam bentuk swimlane

Diagram ini menunjukkan aliran aktivitas yang dijelaskan dengan *use-case* dan pada waktu yang sama menunjukkan aktor yang bertanggung jawab terhadap suatu aksi (juga akan berguna untuk analisis kelas)



Terima kasih

Acknowledgement

- Pengembangan dari slide: "Scenario Based Modeling" IF2036 Sem II - 2012/2013
- Pengembangan dari slide: "Pemodelan Berbasis Skenario" oleh Bayu Hendradjaya, IF2250 Sem II - 2015/2016