Pengembangan Aplikasi Perangkat Lunak

OOAD

Teknik Analisis Requirement & Use Case

Tujuan Pertemuan

- Mengetahui teknik analisis requirement
- Mampu melakukan analisis requirement
- Paham kegunaan use-case diagram
- Mampu membuat use-case diagram

Unified Modeling Language (UML)

- UML → Bahasa Pemodelan Terpadu
- UML → dikembangkan sebagai bahasa untuk pemodelan sistem berorientasi objek.
- Sekarang UML juga digunakan sebagai spesifikasi sistem

Unified Modeling Language (UML)

- Sekarang UML juga menjadi ***alat standar bagi software engineer dan sistem engineer***.

- Melalui diagram UML, dengan menggunakan software khusus ***dapat dihasilkan suatu deskripsi data dan bahkan kode program secara otomatis***.

Unified Modeling Language (UML)

- UML pertama kali dikembangkan oleh; ***Rational Software Company*** , dengan cara memadukan beberapa object-oriented modeling methods:
  - **Booch**

    by: Grady Booch,
  - **OMT** (Object Modeling Technique)

    by: Jim Raumbaugh, and
  - **OOSE** (Object-Oriented Software Engineering),

    by: Ivar Jacobson.
- Selanjutnya Tim ini tergabung dalam OMG (Object Management Group).

UML - Pemodelan Perangkat Lunak

- UML digunakan untuk ***pemodelan sistem perangkat lunak***,
- Pemodelan tersebut mencakup;
  - ***Analisis,*** dan
  - ***Disain.***

UML - Analisis

- Tahapan dalam Analysa Sistem:
  - Sistem ***digambarkan***/deskripsikan oleh sekumpulan ketentuan/syarat/kebutuhan
  - ***Identifikasi*** bagian-bagian sistem pada tingkat tinggi (dengan visualisasi tingkat tinggi, yang mudah dipahami).
- Untuk ***visualisasi tingkat tinggi*** ini digunakan use case.
- ***Use Case*** digunakan untuk:
  - Mengidentifikasi ketentuan/kebutuhan, dan
  - Menentukan ketentuan/kebutuhan

UML - Disain

- Fase disain sangat terkait dengan fase analisis
- Tahapan dalam disain:
  - ***identified system parts***,
  - ***detailed specification*** of these parts

    Untuk visualisasi hasil identifikasi dan spesifikasi detil gunakan ***Class diagrams*** atau ***component diagrams.***
  - ***parts interaction***.

    Untuk visualisasi interaksi bagian-bagian dari sistem gunakan ***interaction diagrams*** atau ***state chart diagrams***

Basic Building Blocks

- Basic Building Block dari UML adalah tentang sesuatu dan relasinya.
- Pada UML terdapat 9 jenis diagram, namun terdapat 5 diagram yang sering disebut sebagai **core diagrams;**

1. **Use Case Diagrams**
2. **Class Diagrams**
3. **Sequence diagrams**
4. **State chart diagrams**
5. **Activity diagrams**
6. Object diagrams
7. Collaboration diagrams
8. Component diagrams
9. Deployment diagrams

Use Case

Fase dalam Pengembangan System

Phase	Actions	Outcome
Initiation	Membuat / membentuk kebutuhan bisnis	Business documents
Analysis	Interview stakeholders, Eksplorasi lingkungan sistem	Organized documentation
Specification	Analisa aspek rekayasa sistem, membuat konsep sistem	Logical System Model
Implementation	Program, build, unit-testing, integrasi, dokumentasi	Testable system
Testing & Integration	Integrasi seluruh komponen, verifikasi, validasi, instalasi, panduan	Testing results, Working sys
Maintenance	Bug fixes, modifikasi, adaptasi	System versions

Use Case

- Use case diagram digunakan pada ***fase awal*** dalam suatu proyek pengembangan software.
- fokus pada menentukan ***bagaimana pengguna eksternal berinteraksi dengan sistem***, bukan untuk menentukan bagaimana sistem harus menyelesaikan tugas-tugas.
- Menggunakan Use case merupakan cara yang baik untuk ***mengekspresikan kebutuhan fungsional dari sistem perangkat lunak***, use case intuitif dan mudah dimengerti sehingga use case dapat digunakan dalam negosiasi dengan non-programmer.

Sumber-sumber Membuat Kebutuhan Bisnis

- Kebutuhan awal berasal dari user, didapat melalui:
  - Dokumen, such as RFI (Request for Information)/RFP (Request for Proposal)
  - Meetings, laporan-laporan
- Kebutuhan selanjutnya datang dari analis, setelah mempelajari:
  - Batasan dam harga
  - Feasibility (technological, organizational etc)
  - Prototypes
- Kebutuhan yang final akan terbentuk setelah melalui suatu proses iteratif.

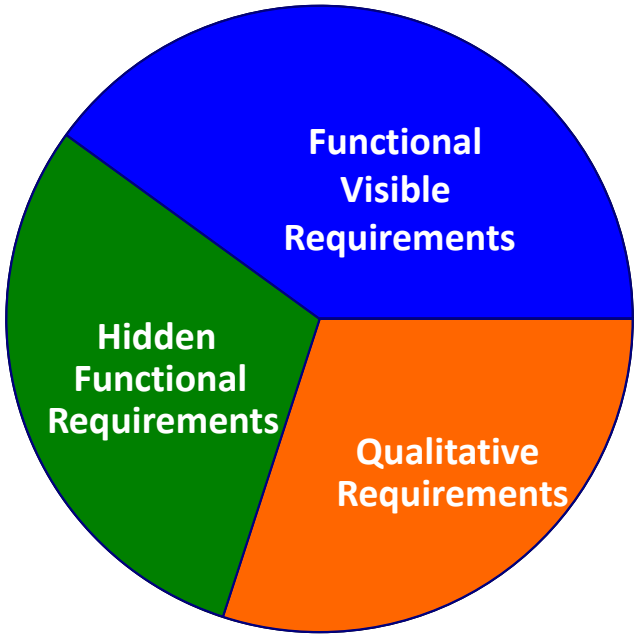
Requirements vs. Design

- Requirements:
  - **What** the system should do
  - More abstract

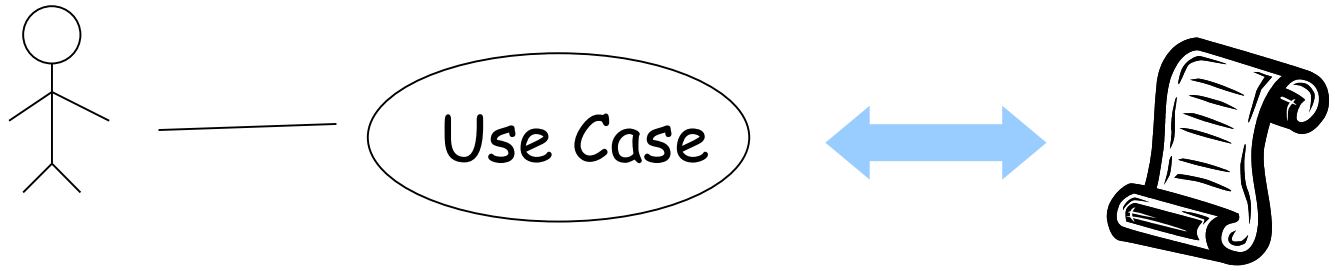
- Design:
  - **How** the system should do it
  - More detailed

Type Requirements

- Visible Functional Requirements
  - "Sistem akan memberikan uang tunai pada pelanggan"
  - "Uang tunai akan disampaikan setelah kartu keluar"
- Qualitative Requirements
  - "proses otorisasi tidak lebih dari 1 detik"
  - "User interface akan mudah digunakan"
- Hidden Requirements
  - "Proses pemeliharaan Database dilakukan setiap malam"



Use-case diagram



Illustration

- Sebuah use case adalah **kontrak** dari interaksi antara ***sistem*** dan ***aktor***.
- Sebuah model use case lengkap terdiri dari :
  - Sebuah diagram yang menggambarkan hubungan antara use-case dan aktor.
  - Sebuah dokumen yang menjelaskan use case dengan rinci.

Objective Menggunakan Use Case

1. Create a semi-formal model of the functional requirements
2. Analyze and define:
   - Scope
   - External interfaces
   - Scenarios and reactions

Use Case digunakan sebagai Sarana Komunikasi

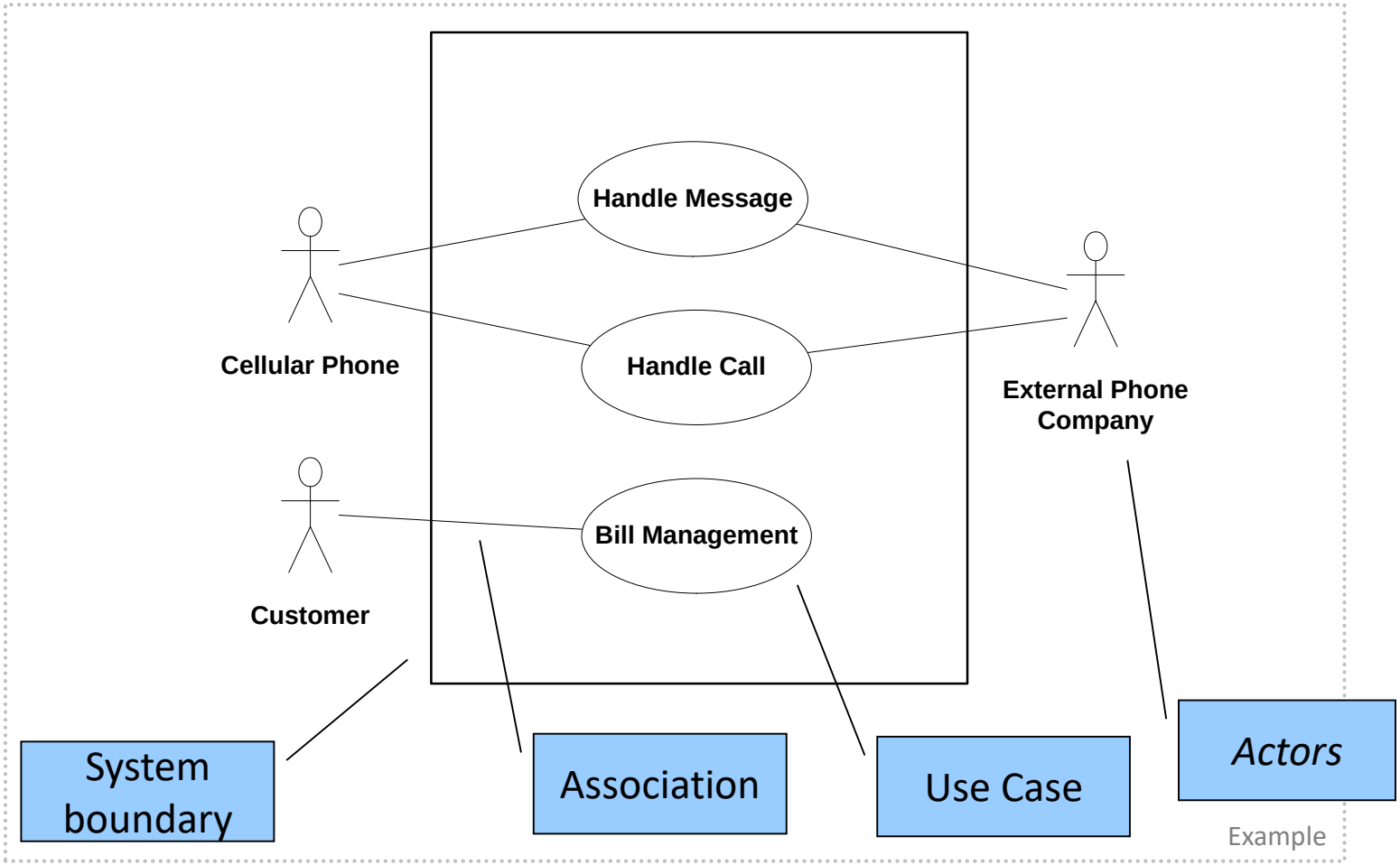
Customers

Designers

Users

- Use case harus menstimulasi diskusi tentang apa yang harus dilakukan sistem, terutama dengan orang-orang yang berada di luar tim pengembangan.

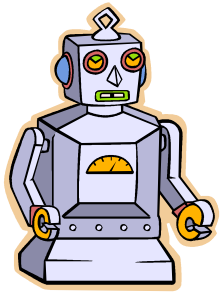
Contoh sederhana



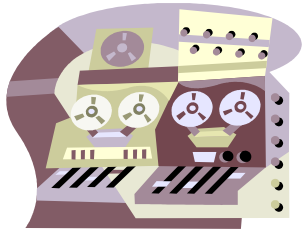
Menentukan Aktor

- Obyek eksternal yang menghasilkan/menggunakan data:
  - Harus berfungsi sebagai sumber dan tujuan untuk data
  - Harus di luar sistem

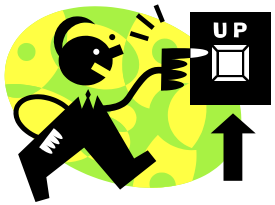
Humans



Machines

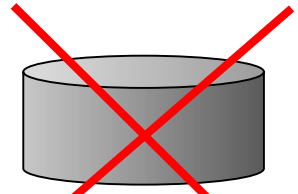


External systems



Sensors

Organizational Units



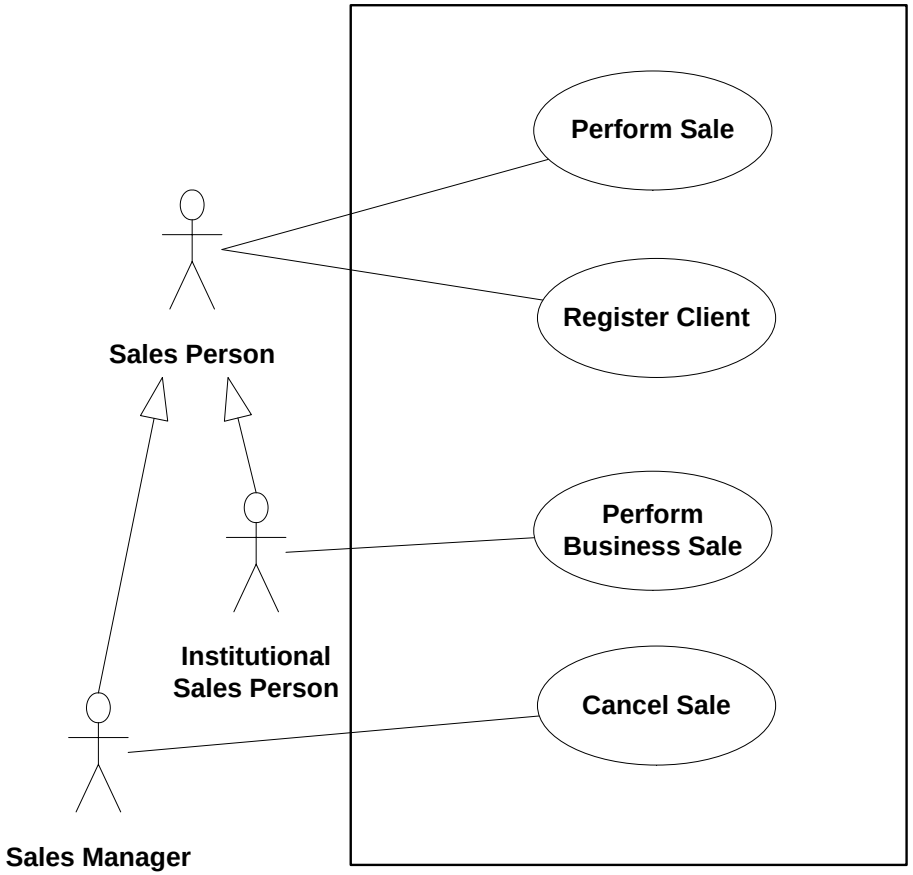
Database

Printer

Aktor dapat digeneralisasi

Aktor turunan (anak) mewarisi seluruh asosiasi use case dari induknya

Harus digunakan jika (dan hanya jika), aktor tertentu memiliki tanggung jawab lebih daripada yang umum (yang digeneralisasi)
(misalnya, berhubungan dengan lebih banyak use-case)



Writing Use-case

Struktur Spesifikasi Use Case

- Name
- Actors
- Trigger *(lihat slide selanjutnya)*
- Preconditions *(lihat slide selanjutnya)*
- Post conditions *(lihat slide selanjutnya)*
- Success Scenario
- Alternatives flows

Alistair Cockburn
"Writing Effective
Use Cases"

Isi Trigger

- Apa yang memulai (trigger) use-case?
- Contoh:
  - Customer melaporkan suatu claim
  - Customer memasukkan card
  - Jam pada Sistem clock is 10:00

Preconditions

- Kondisi yang diperlukan sistem dan harus dipenuhi sebelum menjalankan use-case

- Contoh:
  - Ada User account
  - User memiliki uang yang cukup pada account-nya
  - Terdapat kapasistas penyimpanan yang cukup.

Post conditions

- post-condition adalah outcome (hasil/keluaran) dari use-case. Ada dua kondisi post-condition; **minimal guarantee** dan **success guarantee**.
- Contoh
  - Uang ditransfer ke account user
  - User sudah login (masuk ke sistem)
  - File sudah tersimpan ke hard-disk
- Minimal guarantee
  - Hal minimum yang dapat dijanjikan sistem, menahan even disaat eksekusi use case berakhir failure.
  - **contoh**: Uang tidak ditransfer kecuali otorisasi diberikan oleh user.
- Success guarantee
  - Apa yang terjadi setelah use-case berjalan dengan sukses.
  - Contoh: File disimpan; Uang ditransfer

Success Scenario

- The success scenario adalah alur cerita utama dari use-case
- Ditulis dengan asumsi bahwa segala sesuatunya sudah okay, no errors atau tidak ada problems, dan success scenario mengarah langsung ke hasil yang diinginkan dari use-case.
- Success scenario terdiri dari urutan langkah-langkah
- Contoh:

1. User memasukkan nama produk, SKU and deskripsi
2. System melakukan validasi SKU produk SKU
3. System add (menambahkan) produk ke DB dan menampilkan pesan konfirmasi

Interaction step

Validation Step

Internal Change Step

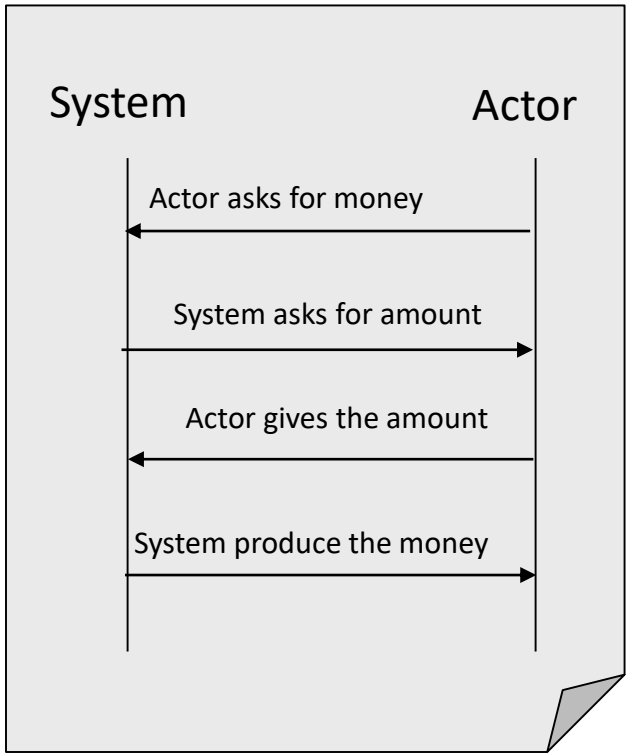
(plus) Interaction Step

Success Scenario Example

1. User memasukkan keyword
2. System menampilkan sekumpulan hasil pencarian, dan produk sponsor, semuanya disertai nama, deskripsi singkat dan harganya.
3. User memilih satu produk
4. Sistem menampilkan halaman produk, termasuk informasi produk, review dan produk terkait.
5. User menambahkan produk ke shopping cart

Panduan untuk Penulisan yang Efektif

- Gunakan gramar yang simple
- Hanya satu pihak/sisi (system or actor) yang melakukan sesuatu pada satu step
- Tulis dari sudut pandang objective
  - Bad: "Get the amount form the user and give him the money"
- Setiap langkah harus mengarah pada beberapa kemajuan
  - Bad: "User click the enter key"



Steps – cont'd

- Penulisan Percabangan (Branches):
  - **If** user memiliki lebih dari $10000 pada account-nya, sistem menyajikan daftar iklan
  - **Otherwise**…

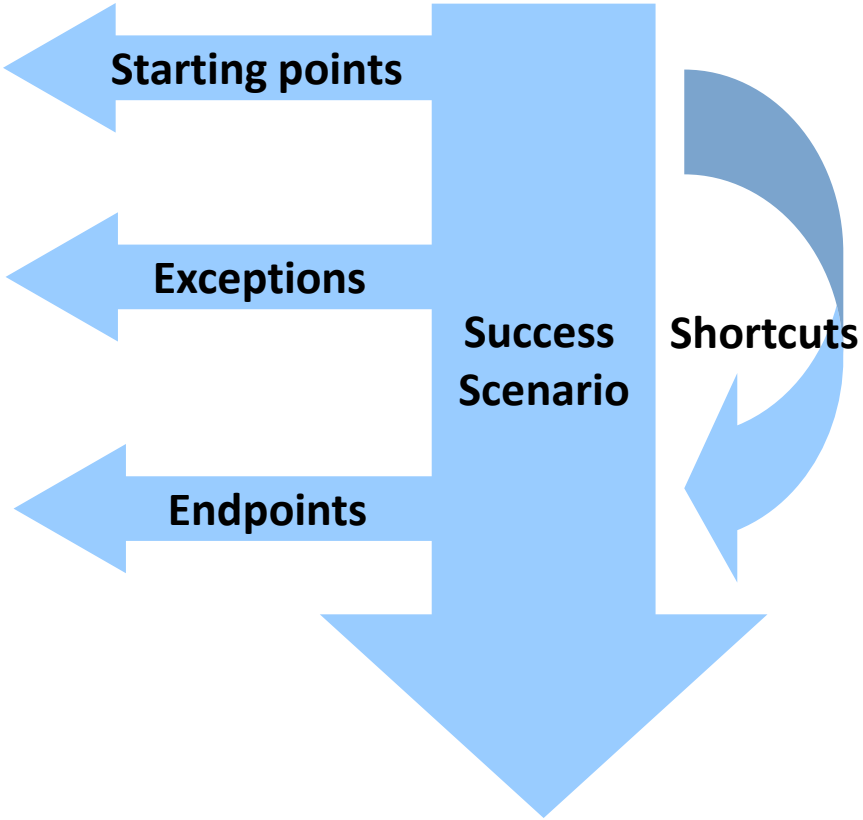
- Penulisan Perulangan (Repeats):
  1. User memasukkan nama item yang ingin dibeli
  2. Sistem menampilkan item
  3. User memilih item untuk dibeli
  4. Systems menambahkan item ke shopping cart
  5. User **repeats steps 1-4 until** indicating he is done

Use-Cases – Common Mistakes

- Diagramnya kompleks
- No system
- No actor
- Terlalu banyak detil user interface
  - "User mengetik ID dan Password, clicks OK atau tekan Enter"
- Terlalu sedikit detil tujuan
  - User mengisi nama
  - User mengisi alamat
  - User mengisi nomer telepon
  - …

Alternative Flows

- Used to describe exceptional functionality
- Examples:
  - Errors
  - Unusual or rare cases
  - Failures
  - Starting points
  - Endpoints
  - Shortcuts



Alternative Flows - Example

- Errors:
  - "jika tidak eject dengan benar"
  - "ada network error terjadi pada langkah 4-7"
  - "segala jenis error yang terjadi"
- Unusual atau kasus yang jarang
  - "Credit card dinyatakan di curi"
  - "User memilih untuk menambah kata baru pada kamus
- Endpoints
  - "Sistem mendeteksi tidak ada lagi isu atau permintaan yang dibuka"
- Shortcuts:
  - "User dapat meninggalkan use-case dengan click pada tombol "esc"

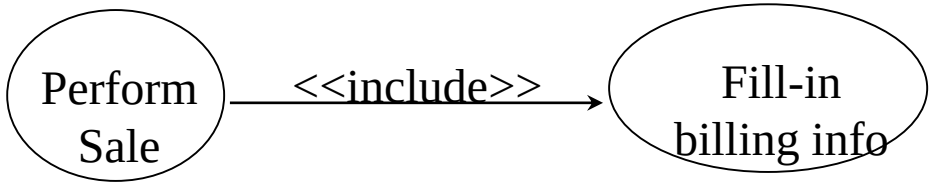
Linking Use-case

Linking Use-case

- Linking memungkinkan flesibilitas pada spesifikasi kebutuhan.
  - Mengisolasi fungsi
  - Memungkinkan sharing fungsi
  - Memecah fungsi menjadi potongan-potongan kecil yang terkelola
- Tiga mekanisme linking yang digunakan:
  - Include
  - Extend
  - Inheritance

Bentuk "Include"

- Include digunakan pada saat:
  - Memecah behaviour (aktifitas/perilaku) yang kompleks
  - Pemusatan behaviour (aktifitas/perilaku) yang umum
  - The base use-case (use-case dasar) secara eksplisit menggabungkan behaviour use-case lain pada lokasi yang ditentukan dalam basis.



Example

Ket:
Perform Sale *termasuk* ***fill-in billing info****,*
Atau
Perform Sale *akan menyertakan* ***Fill-in billing info*** *dalam prosesnya*

Penulisan include

- If a base use-case include another use-case, we will add a reference as a step:

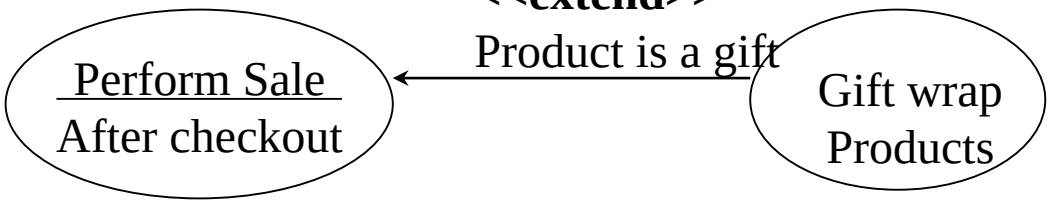
1. System presents homepage
2. User performs login to the system

OR

<include: login to the system>

Bentuk “extend”

- The base use case (use-case dasar) dapat menggabungkan use-case lain pada titik tertentu, yang disebut poin ekstensi.
- Perhatikan arah panah
  - The base use-case tidak tahu use-case mana yang meng-extend



Example

Ket:
***Perform Sale**, jika produk is a gift otomatis **Gift wrap Products** aktif/bekerja, **atau**, **Gift wrap Products** memantau **Perform Sale**, jika Product is a gift Gift wrap Product otomatis aktif*

Penulisan extend

- Scenario tidak menyertakan referensi secara langsung
- Sebaliknya, Scenario menyertakantitik ekstensi, seperti:

  User memasukkan String untuk pencarian
  Sistem menampilkan hasil pencarian

  Extension point: Hasil presentasi

  OR

  <extension point: Hasil presentasi>

- Use case extension menyertakan kondisi dimana ekstensinya dilakukan
  - **Example**: if user termasuk group "rich clients"
  - If lebih dari dua iklan didapatkan

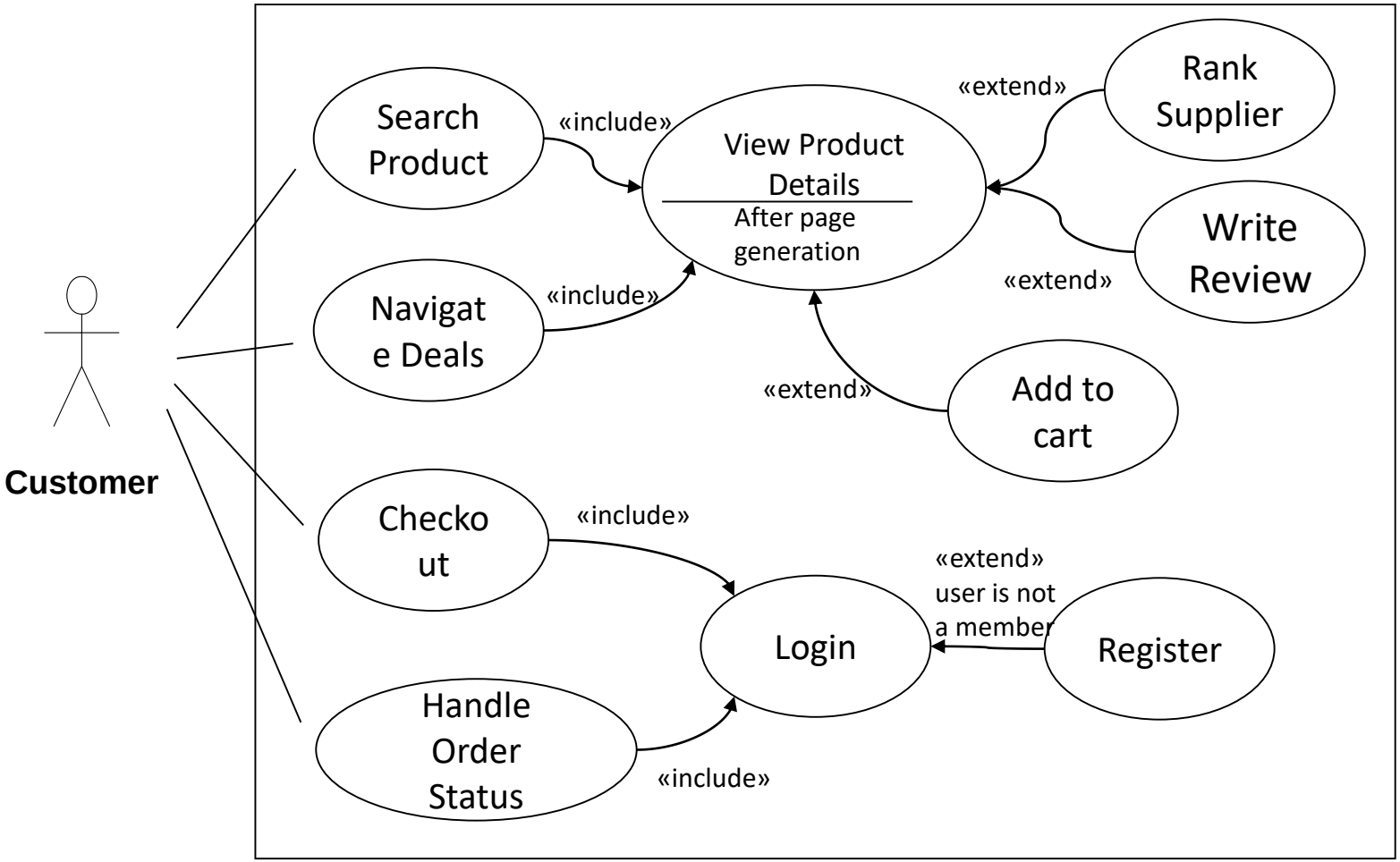
Contoh Amazon

Product Page

Shopping Cart

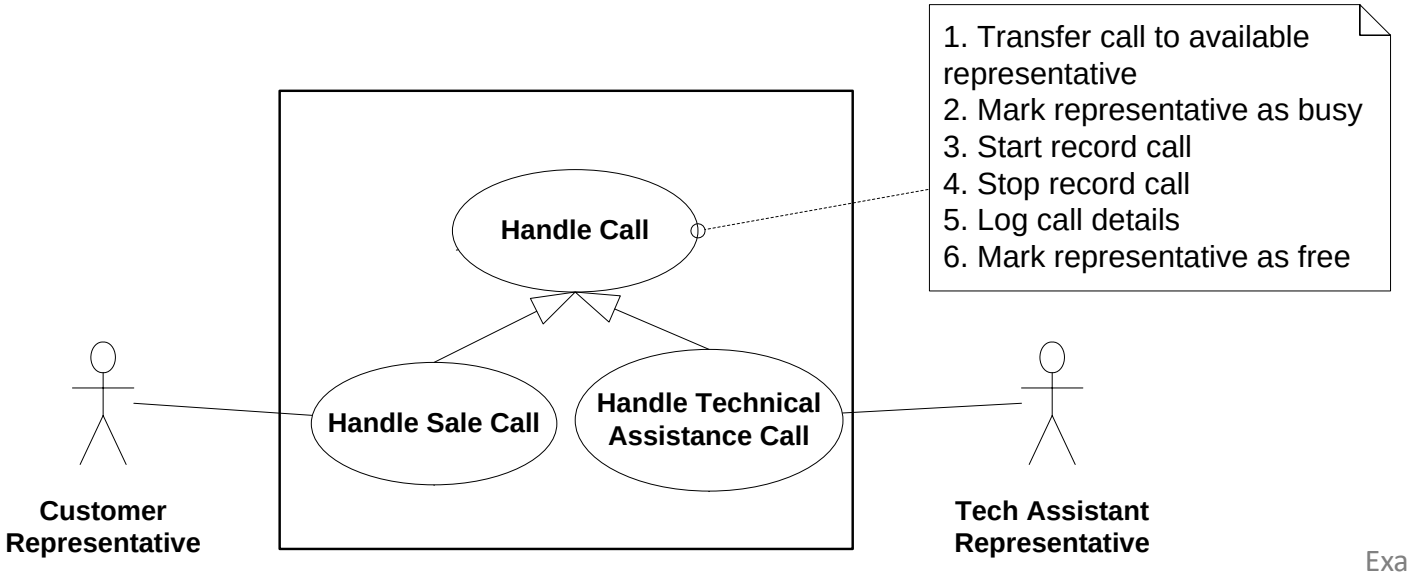
Review Writing Page

Contoh Amazon - lanjutan



Generalization antara Use-Cases

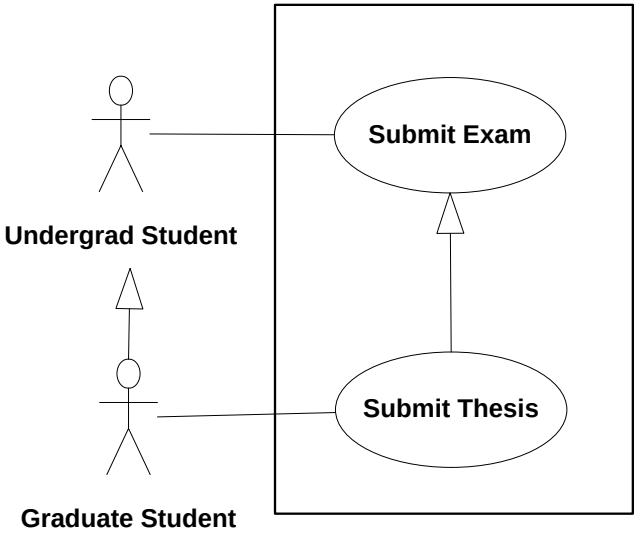
- The child use-case (use-case turunan) menurunkan use-case induk:
  - Interaksi (described in the textual description)
  - Link use-case (associations, include, extend, generalization)
- Child use-case (use-case turunan) dapat menggantikan Use case induk
- Penggantian dilakukan melalui deskripsi textual,

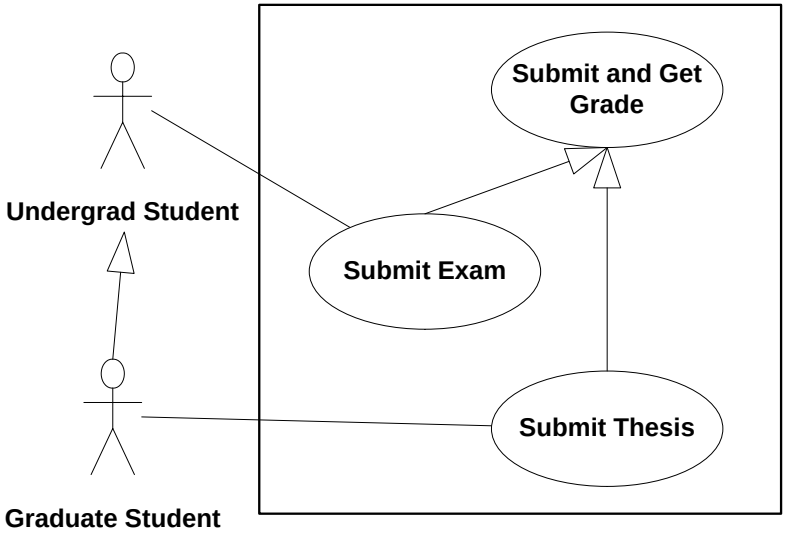


Example

Bahaya generalization

- Mengkombinasi generalization aktor dan use-case dapat berbahaya





Bad: Undergrad can submit thesis

Good: Only graduate student can submit thesis

Contoh: Sistem Operasional Perusahaan telepon

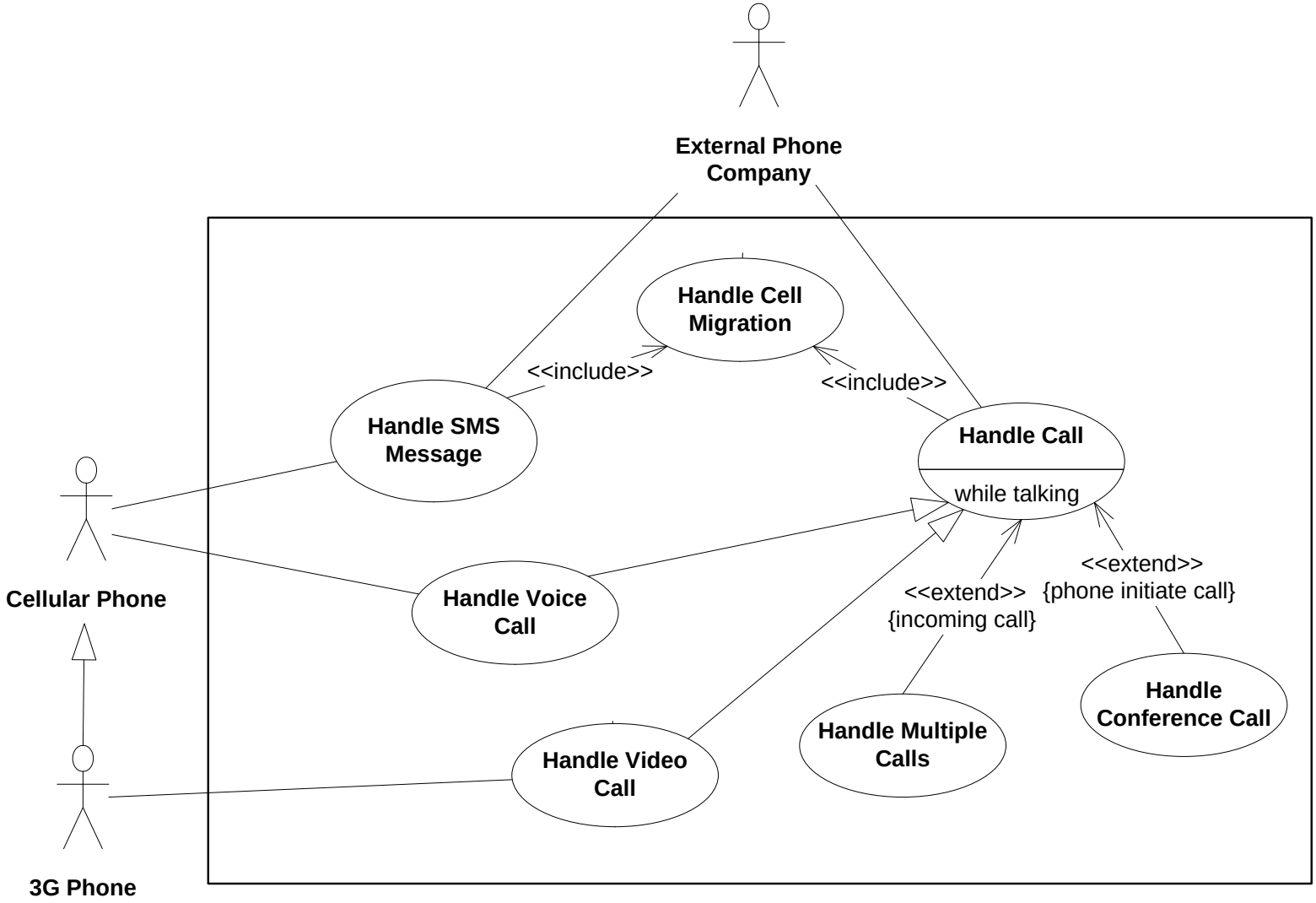
Orange's objective: Membangun sistem yang menangani pesan SMS, menangani panggilan (untuk ponsel generasi 2 dan 3), termasuk panggilan konferensi dan beberapa panggilan dari telepon tunggal. Sistem harus mendukung pengguna bergerak.

Who are the actors?

The Cellular Phone

External Phone companies

Contoh: Sistem Perusahaan Celular

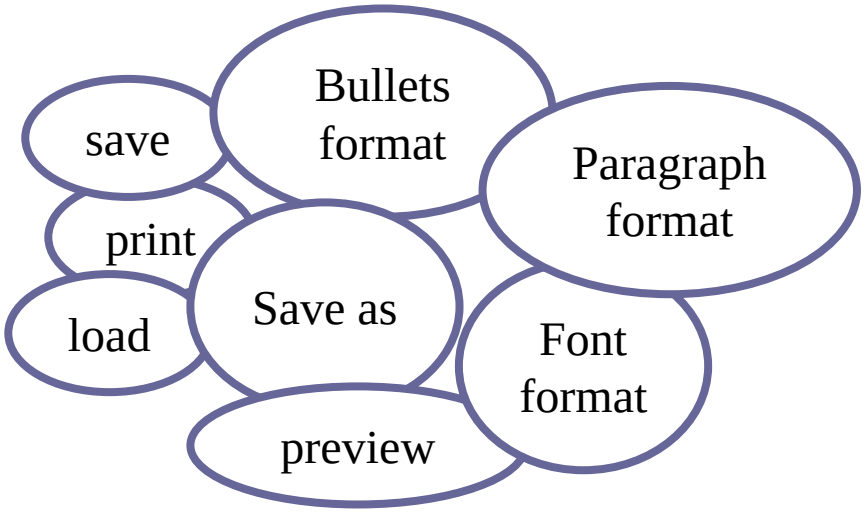


Panduan membuat Use-case yang efektif

How to Model?

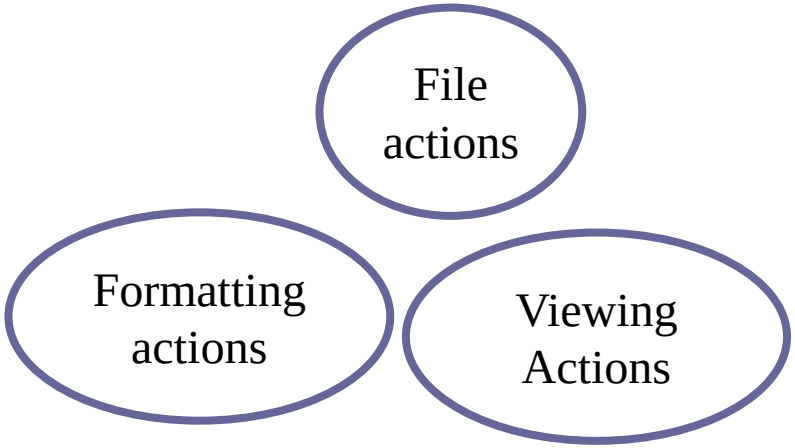
Bottom-up Process

Starting with throwing all scenarios on the page, and then combining them:



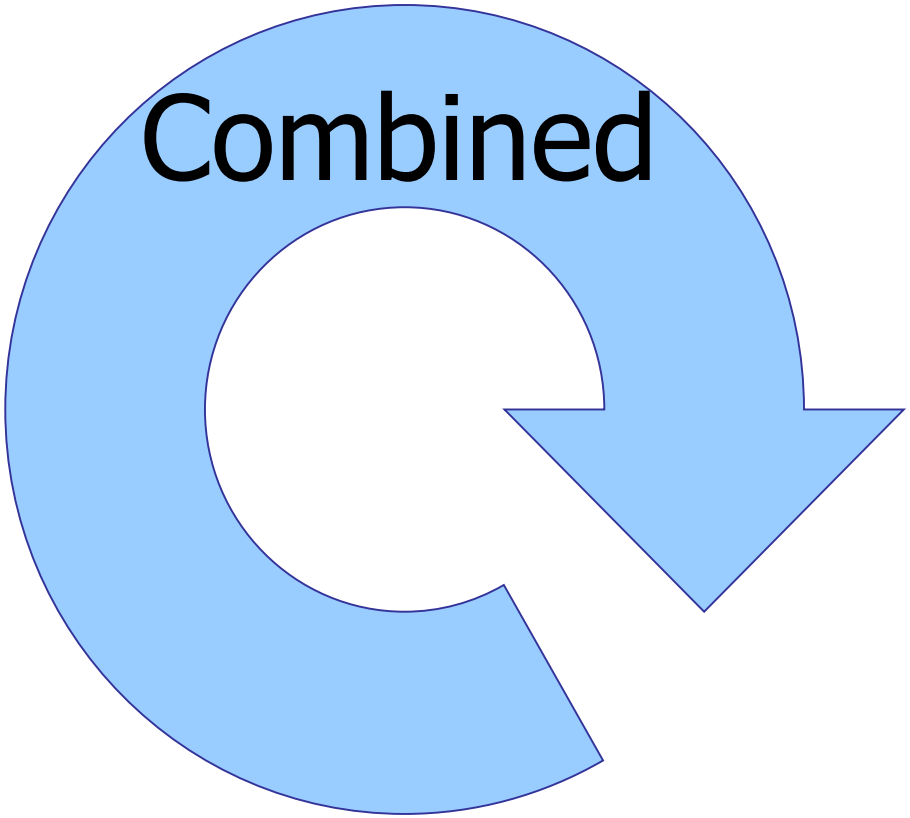
Top-down Process

Starting with an overview of the system, and then splitting Use-cases



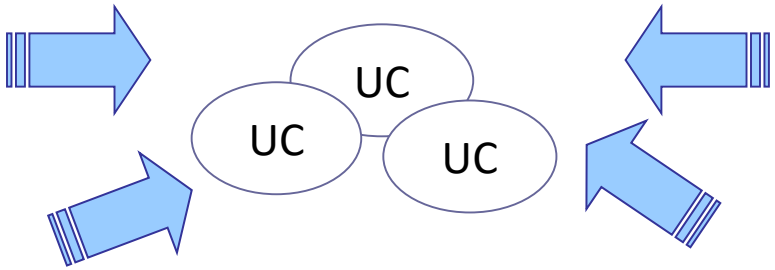
How to Model?

- Kebanyakan proses analisis sebenarnya merupakan:



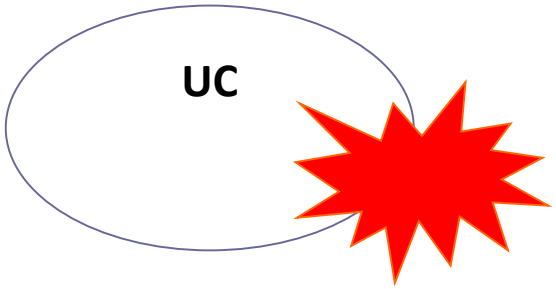
Mengkombinasikan Proses

- **Jumlah batas**:
  - Diagram harus berjumlah antara 3 – 10 base use-case (use-case dasar).
  - Tidak lebih dari 15 use-case (base + included + extending).
- **Abstraksi:**
  - Semua use-case harus memiliki tingkat abstraksi yang se-level
- **Ukuran**:
  - Use-case harus dituangkan dalam setengah halaman atau lebih.
- **Interaksi (interactions)**:
  - Use-case dibuat sebagai bagian dari interaksi yang sama.



Pembagian Proses

- Ukuran:
  - Jika suatu use-case memerlukan lebih dari satu halaman, pertimbangkan menggunakan include/extend
- Ketergantungan lemah (Weak dependency):
  - Jika ketergantungan antara dua bagian use-case lemah mereka harus di pecah.



Panduan tambahan

- Faktor luar yang umum digunakan dan diperlukan oleh beberapa use-case:
  - Jika penggunaannya diperlu gunakan <<include>>
  - Jika base use-case komplit dan penggunaannya hanya optional, pertimbangkan untuk menggunakan <<extend>>
- Diagram use-case harus:
  - Hanya terdiri dari use-case yang memiliki abstrak dengan tingkatan level yang sama
  - Hanya menyertakan aktor yang diperlukan

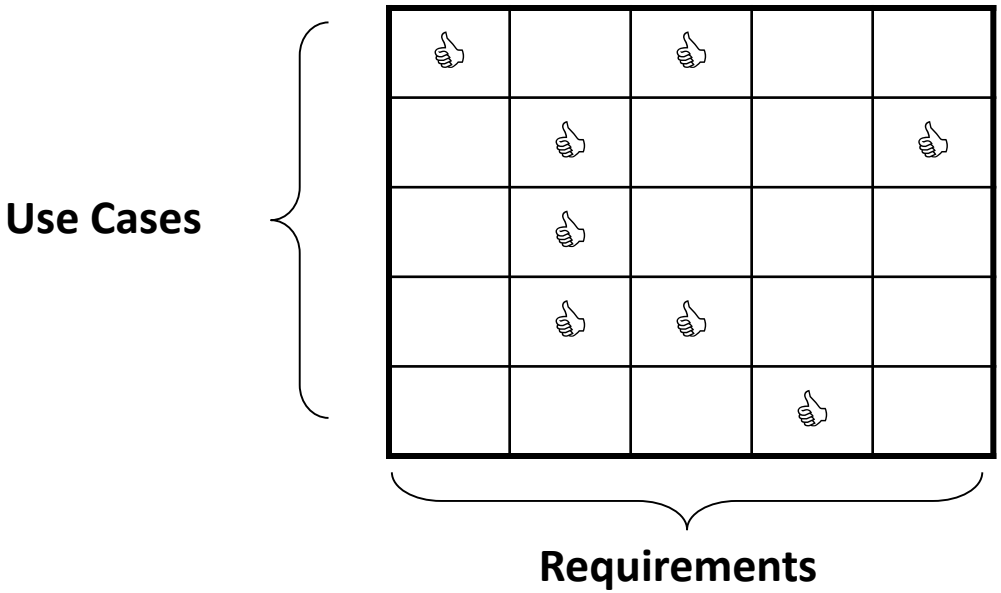
Cakupan

- Cara yang baik untuk menetukan cakupan adalah dengan **in/out lists**:

Topic	In	Out
Setiap bagian non-software pada sistem		✓
Catatan analisis statistik	✓	
Interfacing dengan sistem credit card	✓	
Proses pembersihan Database	✓	
Backup catatan		✓
…		

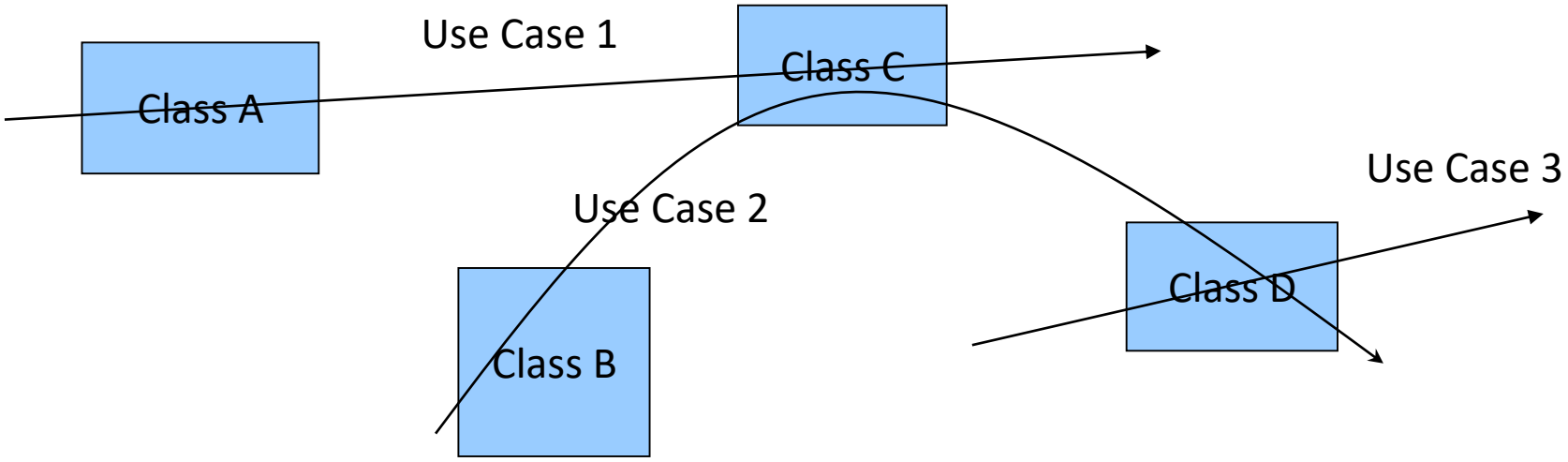
Apa yang Sudah Kita Lakukan

- Jika semua aktor sudah ditentukan.
- Jika setiap kebutuhan fungsi masing-masing memiliki satu use-case yang mengakomodirnya.
- Suatu tabel matrix da[at membantu kita mengetahuisudah sejauh mana kita melakukannya:



Langkah Selanjutnya

- Keluar dan masuknya Data pada sistem direpresentasikan dengan data entities (entitas data) dal **structural diagrams**.
- Behavior yang disebabkan oleh use-cases dapat dilihat pada **behavioral diagrams**.



Ringkasan (Summary)

- ✓ Introduction
  - to the Unified Modeling Language (UML)
  - To Use Case Diagram
- ✓ Use Case Diagrams
  - Dual presentation of use-cases
  - Include, Extend, Inheritance
- ✓ Writing Use Cases
  - Preconditions & Post-conditions
  - Main scenario vs. Alternative Flow
- ✓ Guidelines for Effective Use Cases

Thank's

- See Ya Next Week

Referensi:

- Eran Torch,"*Specifying Requirements with Use Case Diagrams*"

  File: UML-asis04use-case-090707110145-phpapp01.ppt

- Mikael Åkerholm, Ivica Crnković, Goran Mustapić, "*Introduction for Using UML*"

  File: UML-Intro.pdf

- UML, "*Use Case Modeling With Activity And Use Case Diagrams*"

  UML-acticity-UseCase-usecasemodelingnotation-123784689696-phpapp02.ppt