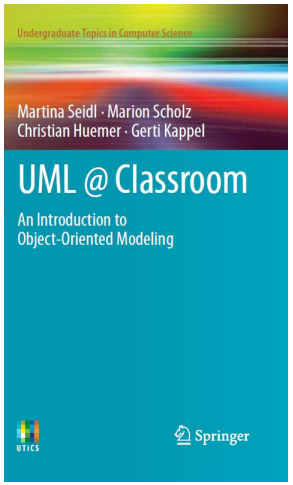
Object-Oriented Modeling

Use Case Diagram



Slide untuk melengkapi buku UML@Classroom
Versi 1.0
Diterjemahkan dari
slide milik Busines Informatics Group, Vienna University of Technology

Program Studi Teknik Informatika
Jurusan Teknik Informatika
Politeknik Negeri Batam
Jalan Ahmad Yani, Batam Center, Batam 29461
www.polibatam.ac.id

Pustaka

- Materi kuliah diambil dari buku berikut:

UML @ Classroom:
An Introduction to Object-Oriented Modeling
Martina Seidl, Marion Scholz, Christian Huemer and Gerti Kappel

Springer Publishing, 2015

ISBN 3319127411

- **Use Case Diagram**
- Structure Modeling
- State Machine Diagram
- Sequence Diagram
- Activity Diagram

Materi

- Pengenalan
- *Use cases*
- *Actors*
- *Relationships* antara *use cases* dan *actors*
- *Relationships* antar *use cases*
- *Relationships* antar *actors*
- Deksripsi *use cases*
- *Best practices*
- Kesalahan umum
- Notasi elemen

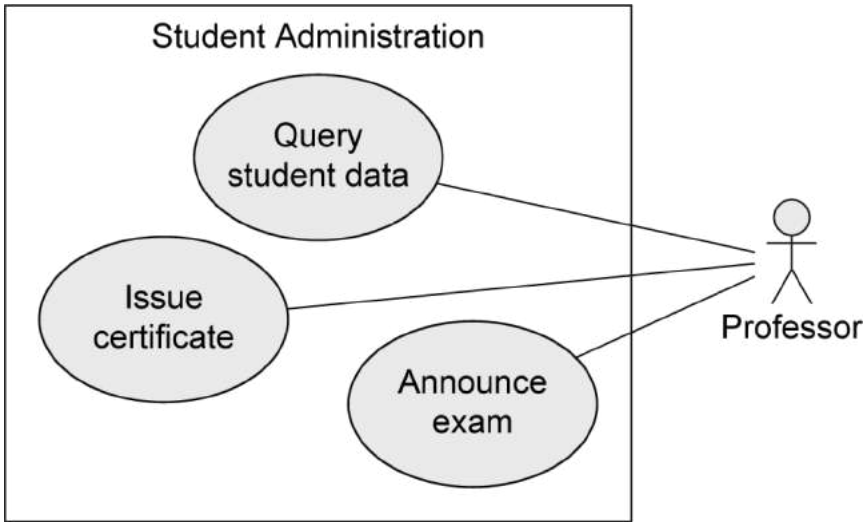
Catatan: materi kuliah ini menggunakan istilah-istilah dalam bahasa aslinya yaitu Bahasa Inggris untuk menghindari kesalahan arti

Pengenalan

- Use case merupakan konsep dasar atau fundamental dalam banyak metode berorientasi objek.
- Use case diagram menggambarkan keinginan atau kebutuhan dari customers/stakeholders
  - sangat penting untuk rancangan detail
- Use case diagram digunakan dalam seluruh proses analisis dan desain.
- Kita bisa menggunakan use case diagram untuk menjawab beberapa pertanyaan berikut:
  - Apa yang digambarkan? (Sistem.)
  - Siapa yang berinteraksi dengan sistem? (*Actors*.)
  - Apa yang bisa dilakukan oleh *actor*? (*Use cases*.)

Contoh: Student Administration System

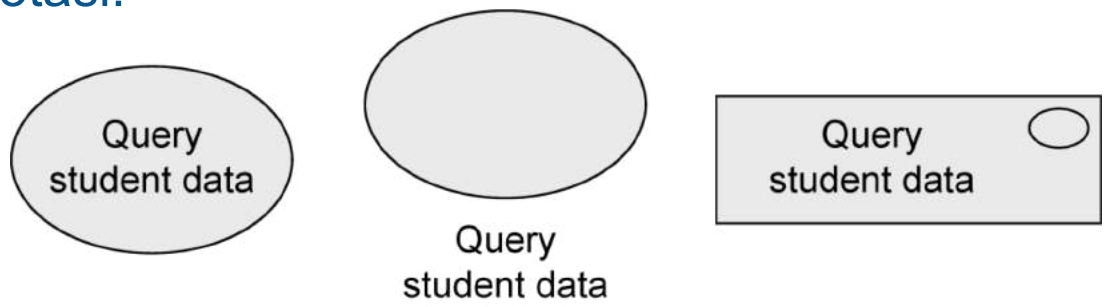
- **Sistem**
  (apa yang digambarkan?)
  - Student administration system
- **Actors**
  (siapa yang berinteraksi dengan sistem?)
  - Professor
- **Use cases**
  (apa yang bisa dilakukan actor?)
  - Query student data
  - Issue certificate
  - Announce exam



Use Case

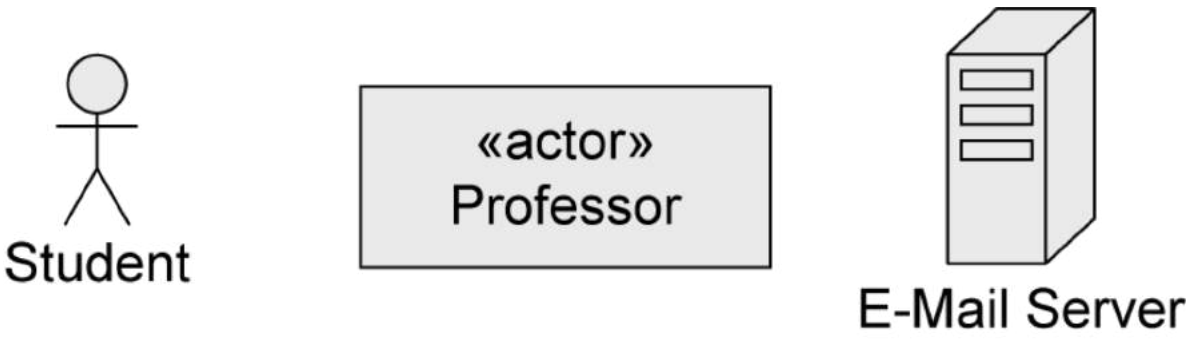
A

- Menggambarkan fungsionalitas yang diharapkan dari sistem yang sedang dikembangkan.
- Memberikan keuntungan nyata untuk satu atau lebih actor yang berkomunikasi dengan use case tersebut.
- Diturunkan dari kumpulan keinginan user.
- Kumpulan dari semua use case menggambarkan fungsionalitas yang akan disediakan oleh sistem.
  - Mendokumentasikan fungsionalitas yang ditawarkan oleh sistem.
- Alternatif notasi:



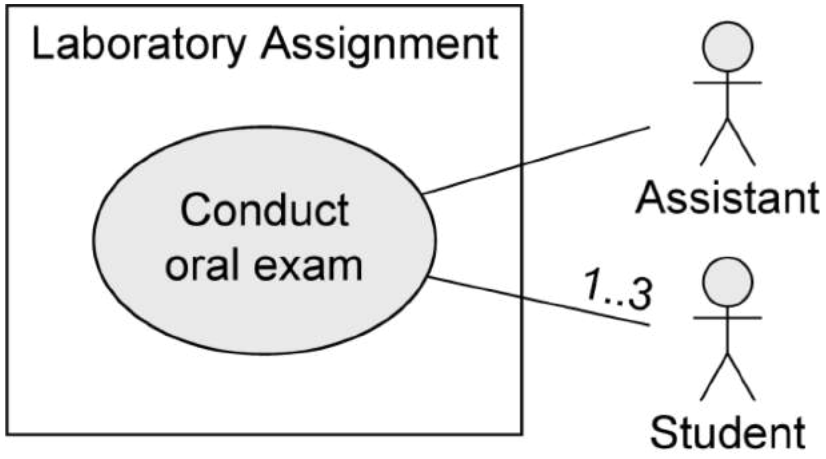
Actor (1/3)

- Actors berinteraksi dengan sistem …
  - dengan **menggunakan** use cases,
    yaitu actors menginisiasi berjalannya sebuah use case.
  - dengan **digunakan** oleh use cases,
    yaitu actors menyediakan fungsionalitas untuk berjalannya sebuah use case.
- Actors melambangkan roles/peran yang dilakukan oleh user.
  - Masing-masing user dapat melakukan beberapa role sekaligus.
- Actors **bukan** bagian dari sistem, yaitu mereka berada di luar batas sistem.
- Alternatif notasi:



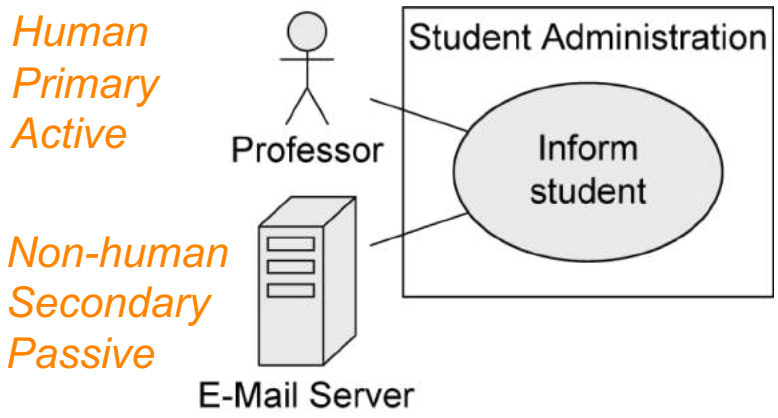
Actor (2/3)

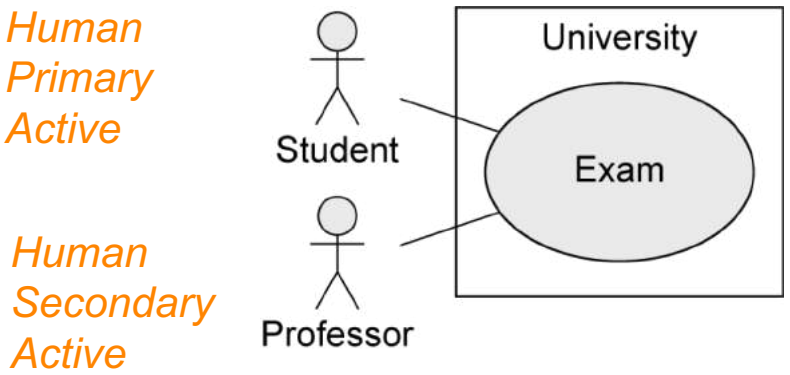
- Biasanya data mengenai user juga dikelola di dalam sistem. Data ini dimodelkan di dalam sistem sebagai object dan class.
- Contoh: actor **`Assistant`**
  - Actor **`Assistant`** berinteraksi dengan sistem **`Laboratory Assignment`** dengan menggunakannya.
  - Class **`Assistant`** menggambarkan object yang merepresentasikan data user (misalnya name, number, …).



Actor (3/3)

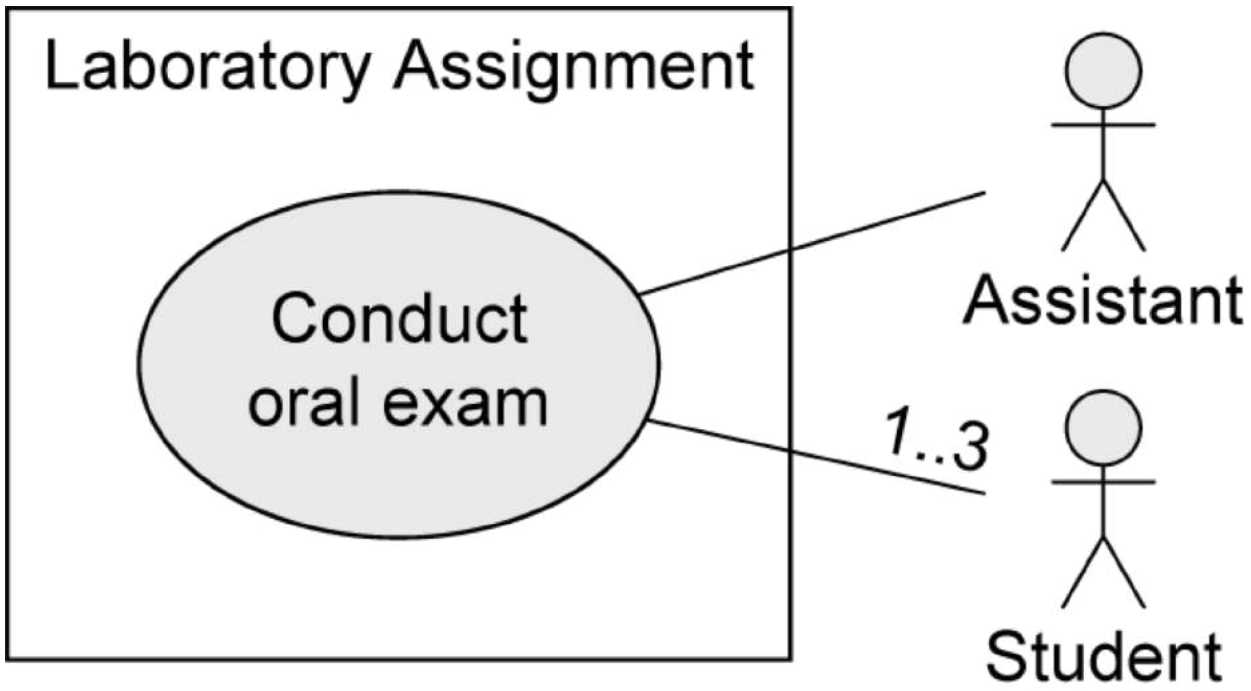
- **Manusia**, misal **`Student`**, **`Professor`**
- **Bukan manusia**, misal **`E-Mail Server`**
- **Primary**: memiliki keuntungan utama terhadap berjalannya sebuah use case
- **Secondary**: tidak menerima keuntungan langsung
- **Active**: menginisiasi berjalannya sebuah use case
- **Passive**: memberikan fungsionalitas untuk berjalannya sebuah use case
- Contoh:



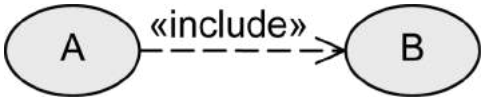


Relationships antara Use Cases dan Actors

- Actors terhubung dengan use cases melalui garis (*associations*).
- Setiap actor harus berkomunikasi dengan **minimal** satu use case.
- Sebuah association selalu binary.
- Multiplicities boleh dituliskan.

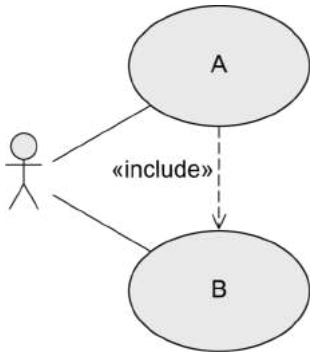


Relationships antar Use Cases



«inlcude» - Relationship

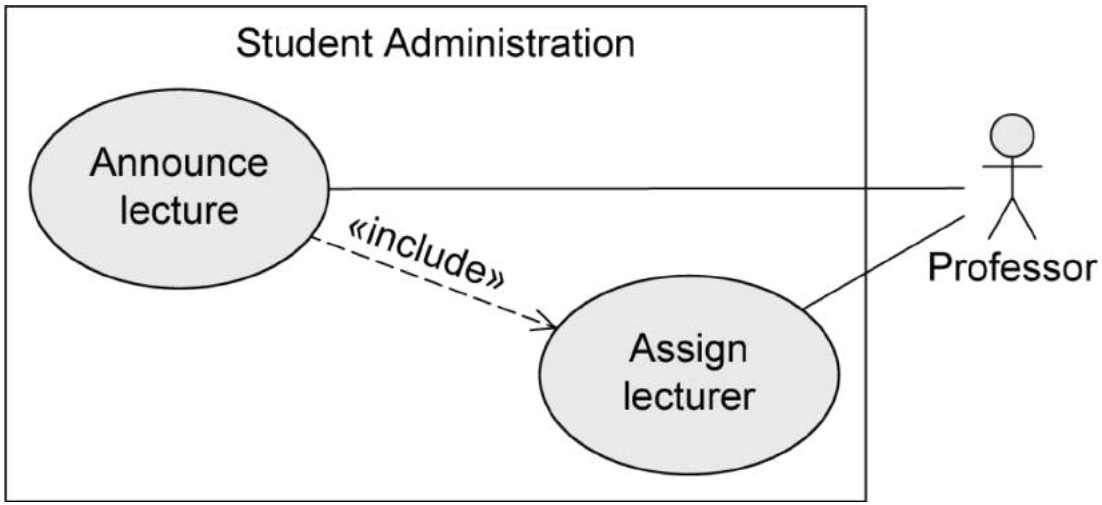
- Perilaku sebuah use case (*included use case*) diintegrasikan ke dalam perilaku use case lain (*base use case*)



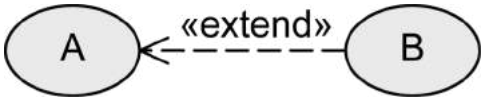
Base use case
memerlukan perilaku dari *included use case* agar dapat menjalankan fungsionalitasnya

Included use case
dapat dijalankan secara terpisah

- Contoh:

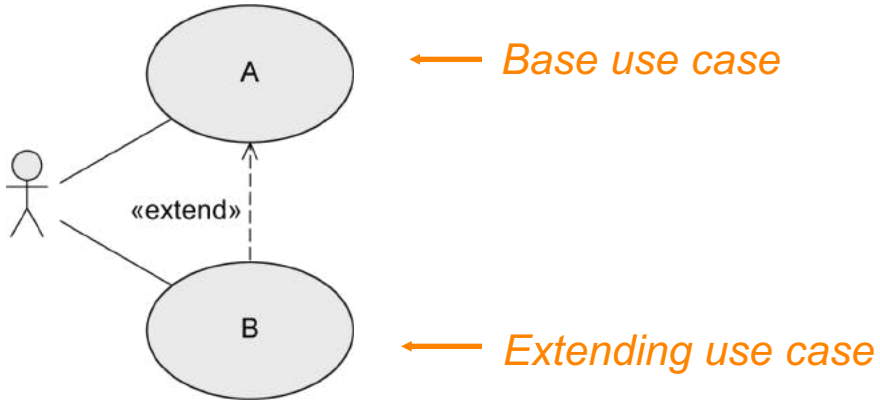


Relationships antar Use Cases



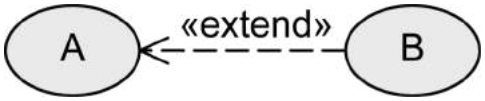
«extend» - Relationship

- Perilaku sebuah use case (*extending use case*) dapat diintegrasikan ke dalam perilaku use case lain (*base use case*) tapi tidak harus.
- Kedua use case dapat dijalankan sendiri-sendiri secara terpisah.



- A memutuskan apakah perlu menjalankan B.
- Extension points mendefinisikan dimana perilaku diintegrasikan.
- Conditions mendefinisikan syarat yang harus dipenuhi agar perilaku dapat diintegrasikan.

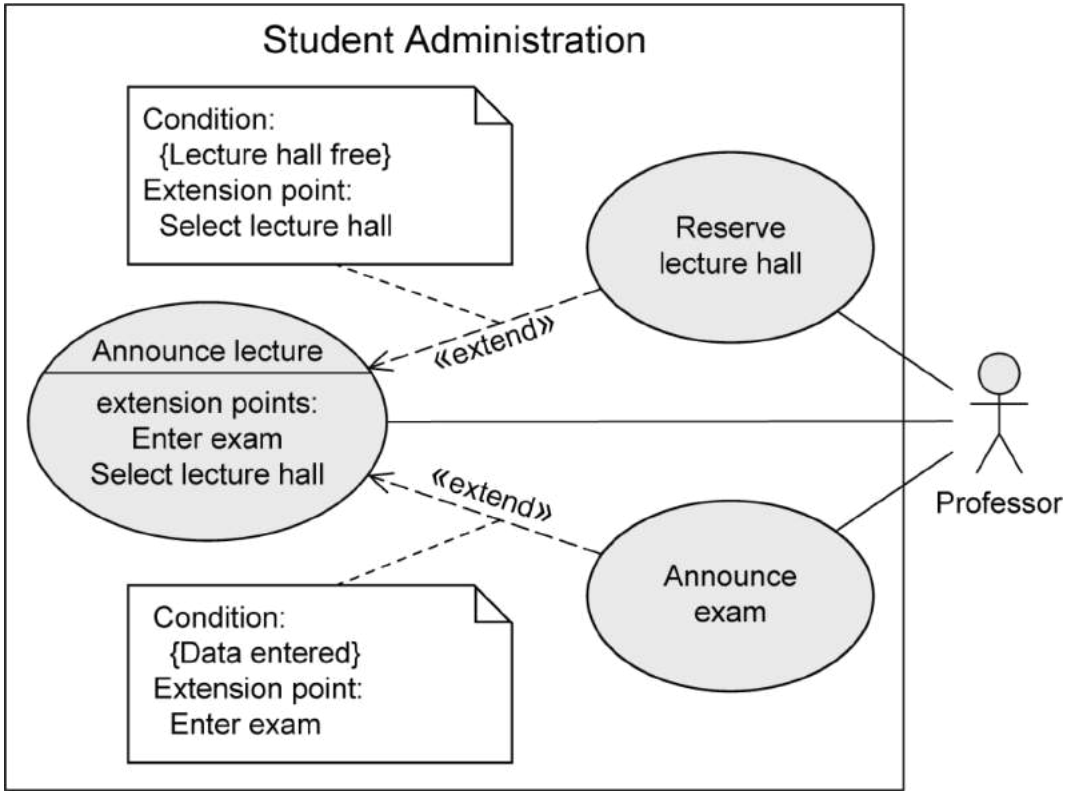
Relationships antar Use Cases



«extend» - Relationship: Extension Points

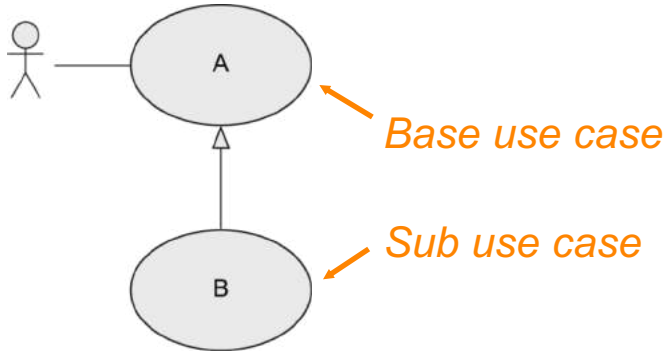
- Extension points dituliskan di dalam use case.
- Memiliki beberapa extension points diperbolehkan.

- Contoh:

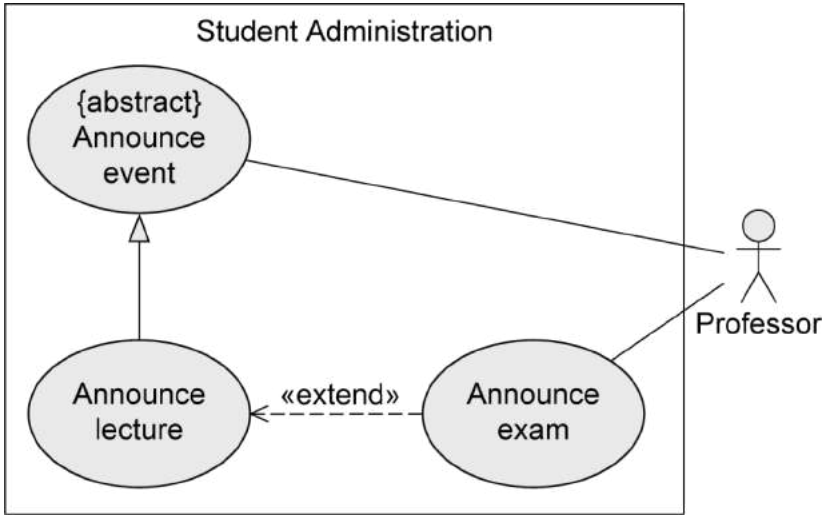


Relationships antar Use Cases

Generalization of Use Cases

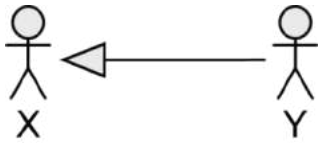


- Use case **A** meng-generalisasi use case **B**.
- **B** mewarisi perilaku **A** dan mungkin menambahkan atau mengubahnya.
- **B** juga mewarisi semua relationships **A**.
- **B** melakukan semua fungsionalitas **A** tapi memutuskan sendiri bagian mana dari A yang dijalankan atau diubah.
- **A** dapat dilabeli sebagai **{abstract}**
  - Tidak dapat dijalankan secara langsung
  - Hanya **B** yang dapat dijalankan
- Contoh:

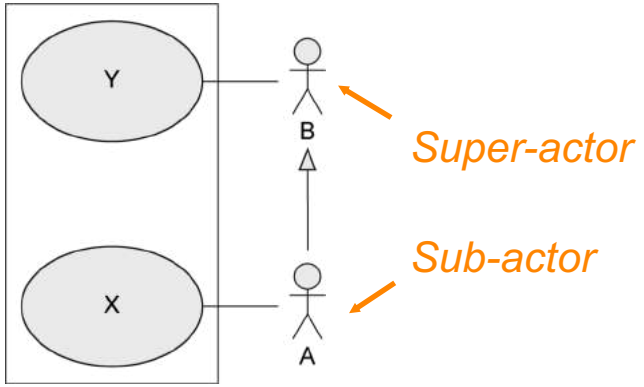


Relationships antar Actors

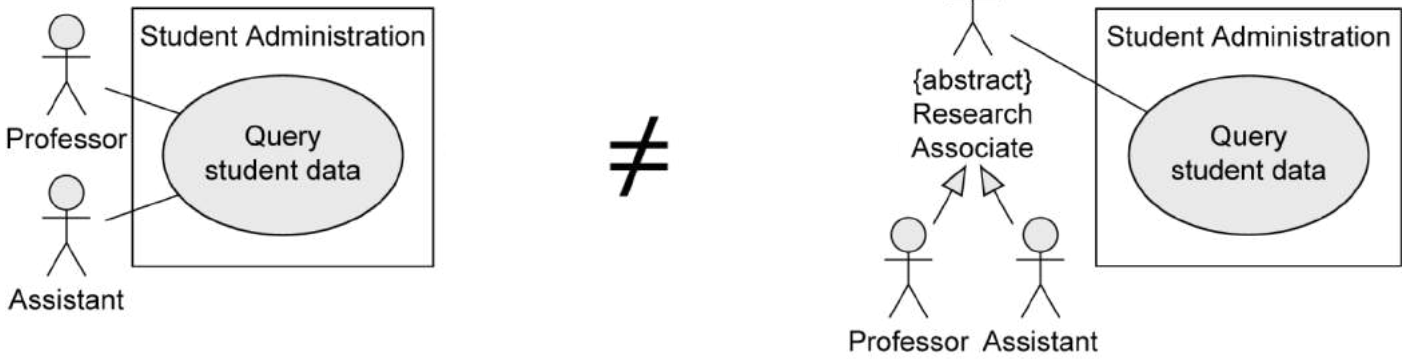
Generalization of Actors



- Actor **A** mewarisi actor **B**.
- **A** dapat berkomunikasi dengan **X** and **Y**.
- **B** hanya dapat berkomunikasi dengan **Y**.
- *Multiple inheritance* diperbolehkan.
- *Abstract* actors diperbolehkan.



- Contoh:



Professor DAN **Assistant** diperlukan untuk menjalankan **Query student data**

Professor ATAU **Assistant** diperlukan untuk menjalankan **Query student data**

Deskripsi Use Cases

- *Structured approach (pendekatan terstruktur)*
  - *Name*: nama
  - *Short description*: deksripsi singkat
  - *Precondition*: syarat agar dapat dijalankan dengan sukses
  - *Postcondition*: keadaan sistem setelah dijalankan secara sukses
  - *Error situations*: kesalahan yang terhubung dengan problem domain
  - *System state in the event of an error*: keadaan sistem pada saat terjadi error
  - *Actors*: actor yang dapat berkomunikasi dengan use case
  - *Trigger*: event yang menjalankan/menghentikan use case
  - *Standard process*: langkah-langkah yang harus dilakukan
  - *Alternative processes*: penyimpangan dari proses standard

  [A. Cockburn: Writing Effective Use Cases, Addison Wesley, 2000]

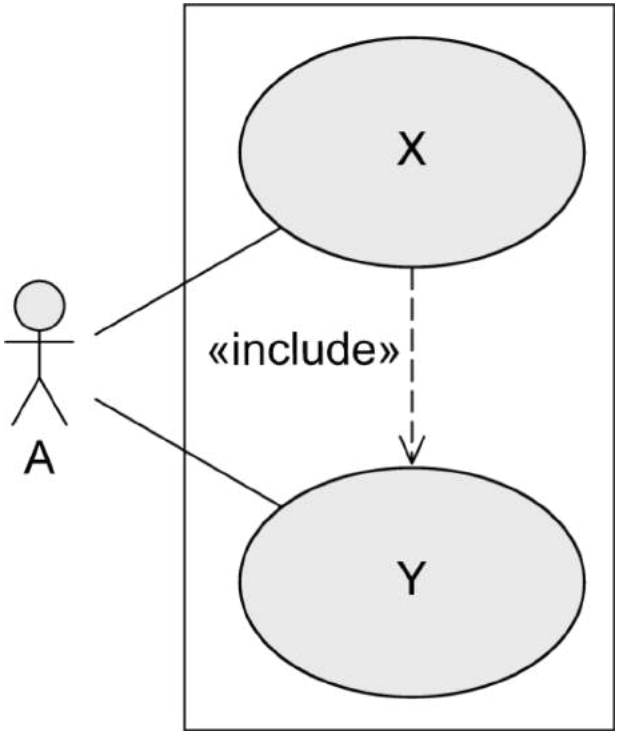
Description of Use Cases - Contoh

- Nama: **`Reservasi ruang kuliah`**
- Short description: Seorang karyawan melakukan reservasi sebuah ruang kuliah di unversitast untuk sebuah event.
- Precondition: Karyawan memiliki autorisasi untuk melakukan reservasi ruang kuliah.
- Postcondition: Sebuah ruang kuliah telah direservasi.
- Error situations: Tidak ada ruang kuliah yang kosong.
- System state in the event of an error: Karyawan tidak berhasil melakukan reservasi ruang kuliah.
- Actors: **`Karyawan`**
- Trigger: Karyawan memerlukan sebuah ruang kuliah.
- Standard process: (1) Karyawan melakukan login ke sistem.
  (2) Karyawan memiliki ruang kuliah.
  (3) Karyawan memilih tanggal.
  (4) Sistem mengkonfirmasi bahwa ruang kuliah tersebut kosong.
  (5) Karyawan mengkonfirmasi reservasi.
- Alternative processes: (4') Ruang kuliah tidak kosong.
  (5') Sistem memberikan alternative ruang kuliah yang lain.
  (6') Karyawan memilih ruang kuliah yang lain dan mengkonfirmasi reservasi.

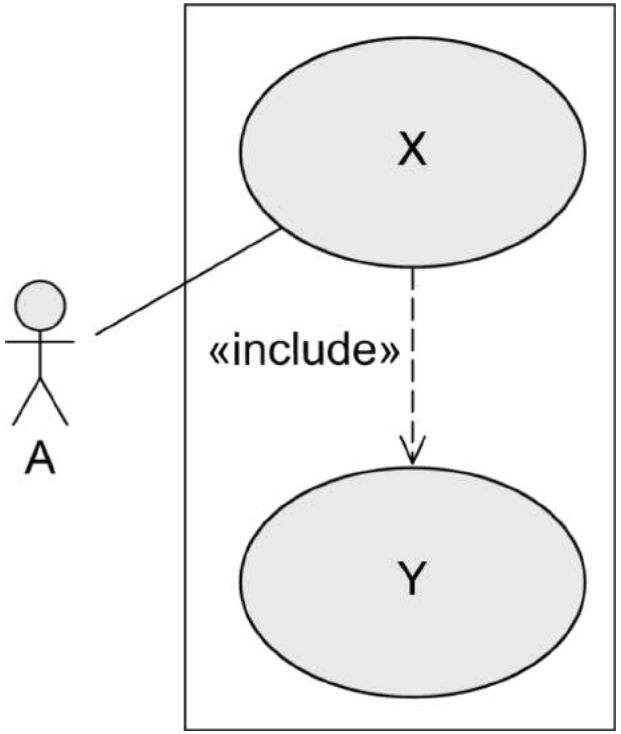
Best Practices

«include»

UML standard



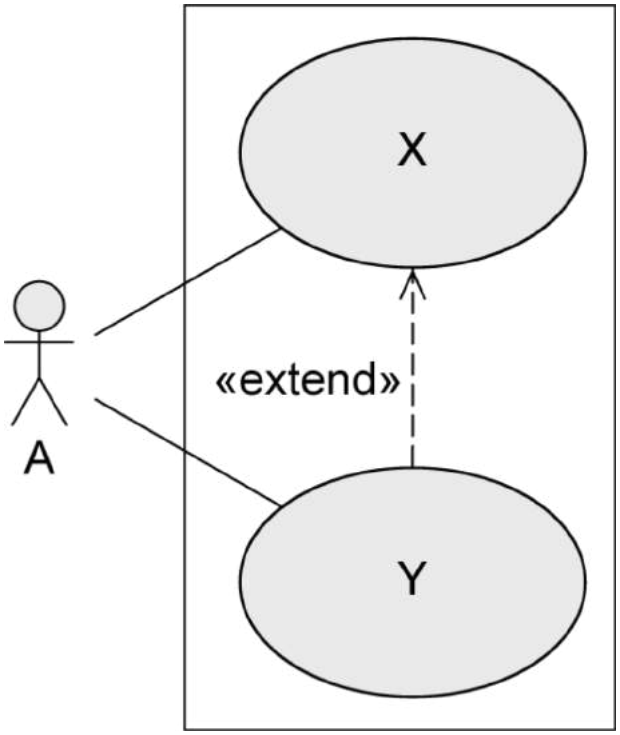
Best practice



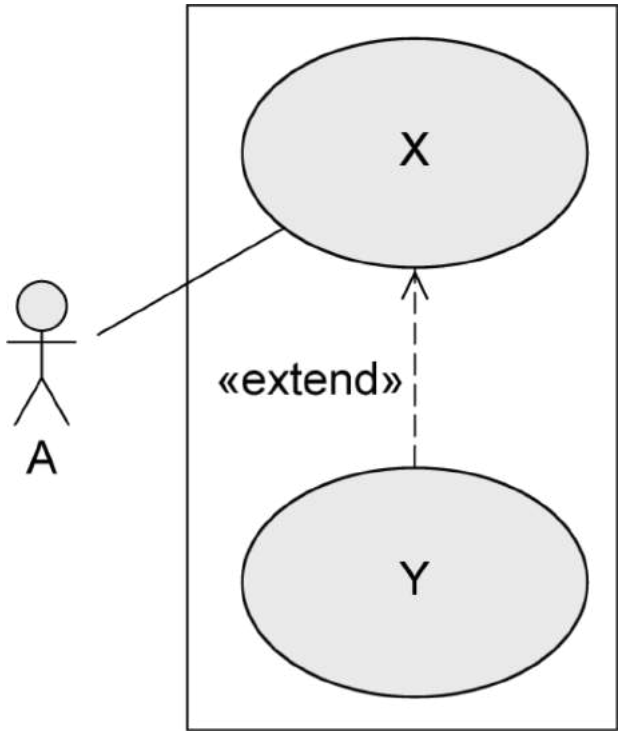
Best Practices

«extend»

UML standard



Best practice



Best Practices

Mengindentifikasi Actors

- Siapa pengguna utama use cases?
- Siapa yang membutuhkan support untuk pekerjaan harian mereka?
- Siapa yang bertanggung jawab terhadap administrasi sistem?
- Sistem harus berkomunikasi dengan *external devices/(software) systems* apa saja?
- Siapa yang berkepentingan dengan hasil sistem?

Best Practices

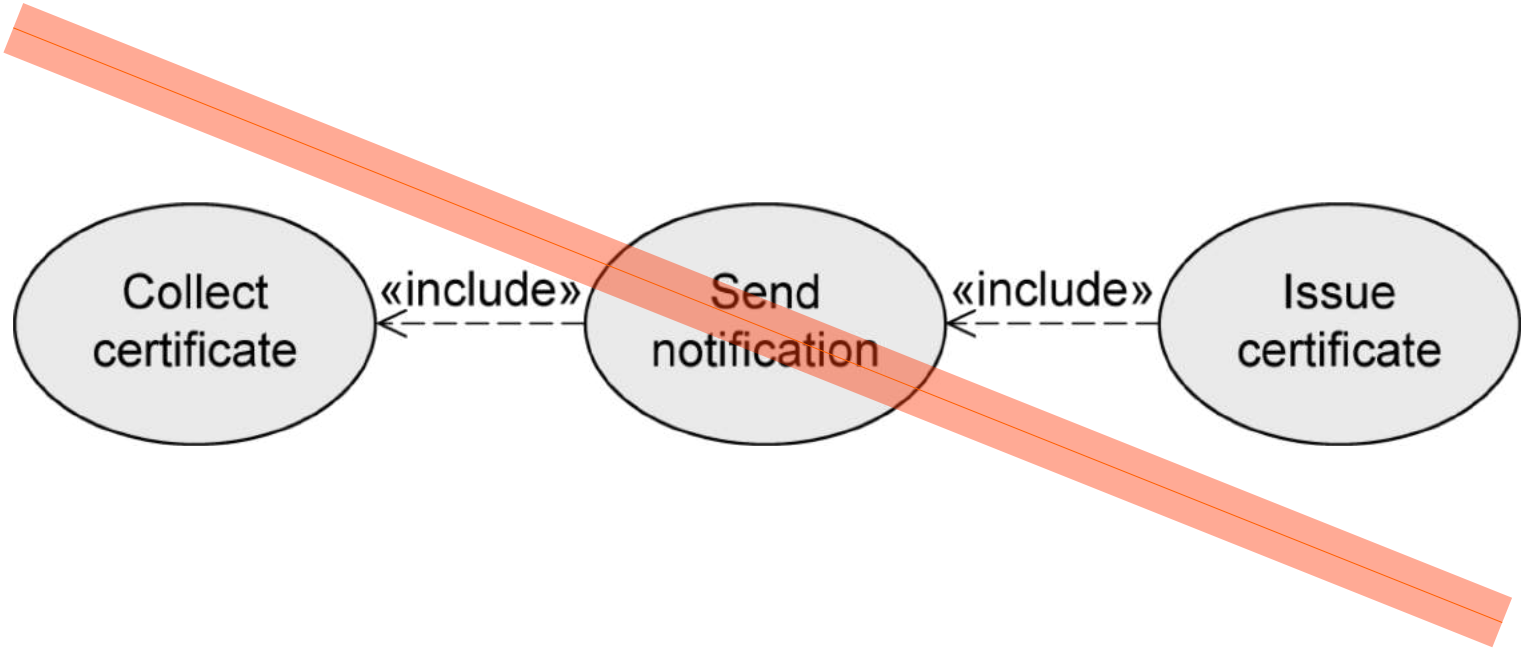
Mengidentifikasi Use Cases

- Apa saja pekerjaan utama yang harus dilakukan seorang actor?
- Apakah seorang actor ingin mendapatkan atau bahkan mengubah informasi yang ada dalam sistem?
- Apakah seorang actor ingin memberitahukan sisten mengenai perubahan yang terjadi di sistem yang lain?
- Haruskah seorang actor diberitahu mengenai event-event yang tidak diharapkan yang terjadi dalam sistem?

Best Practices

Kesalahan Umum yang Harus Dihindari (1/5)

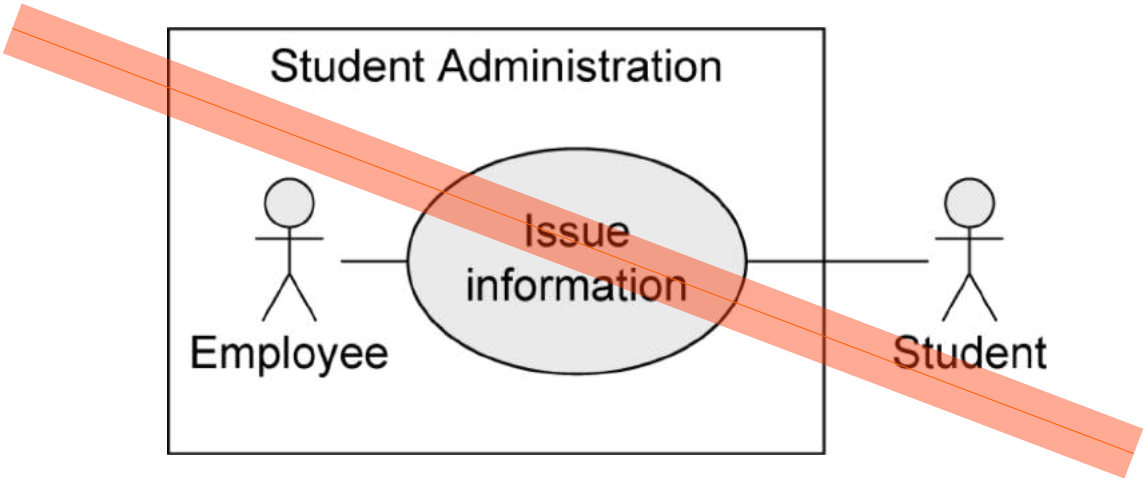
- Use case diagrams **tidak** memodelkan processes/workflows!



Best Practices

Kesalahan Umum yang Harus Dihindari (2/5)

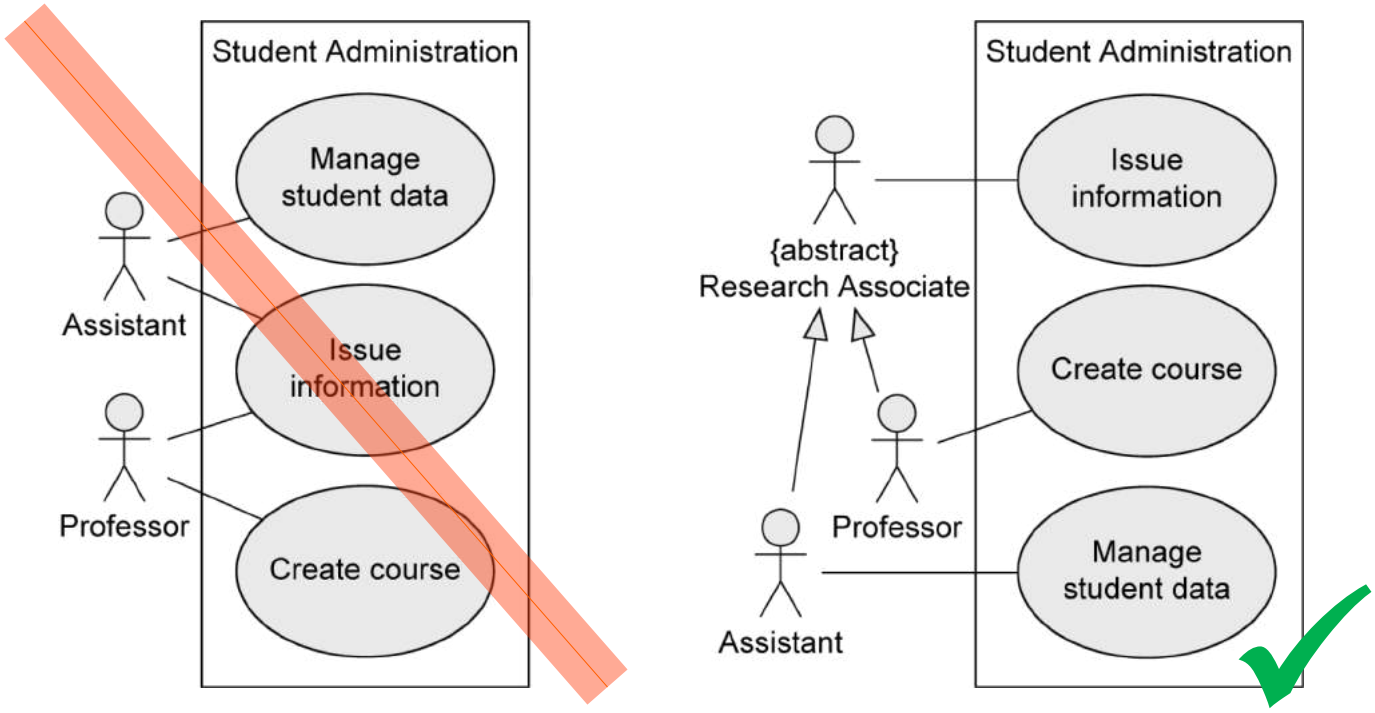
- Actors **bukan** bagian dari sistem, sehingga mereka seharusnya berada di luar batas sistem!



Best Practices

Kesalahan Umum yang Harus Dihindari (3/5)

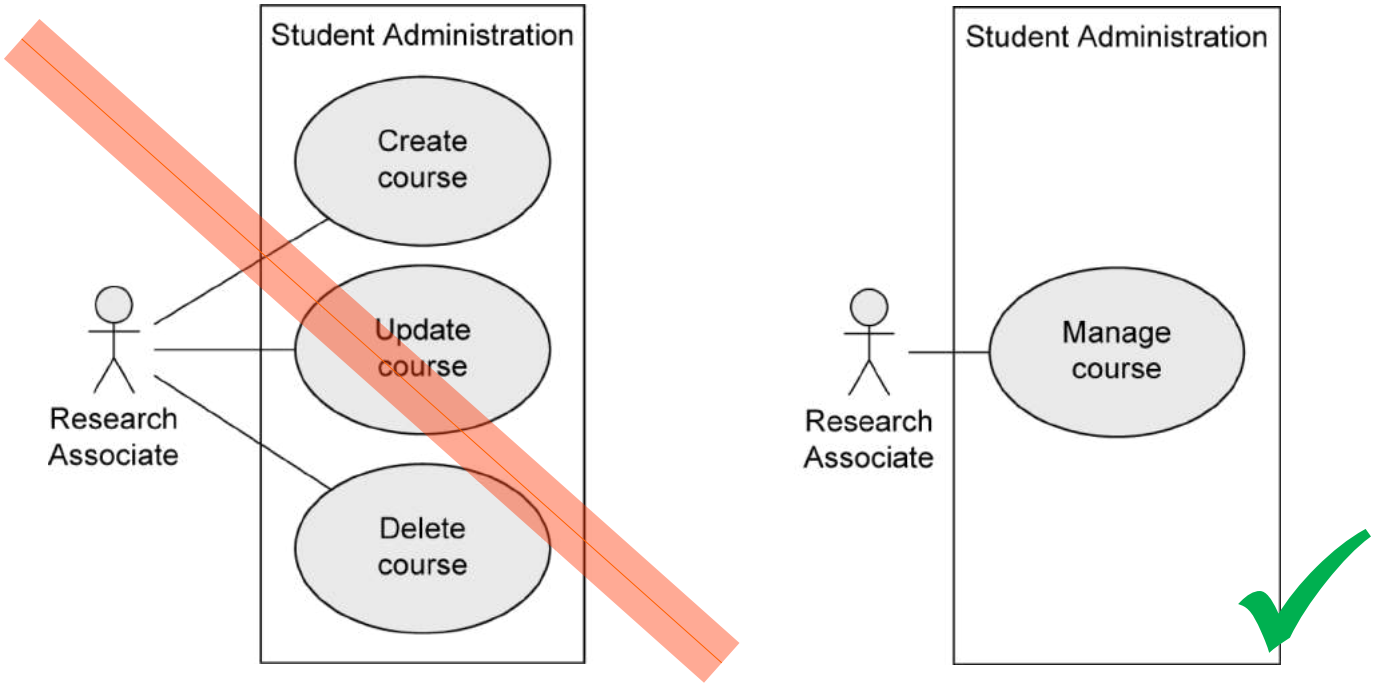
- Use case **`Issue information`** membutuhkan satu actor **`Assistant`** **ATAU** satu actor **`Professor`** untuk berjalan



Best Practices

Kesalahan Umum yang Harus Dihindari (4/5)

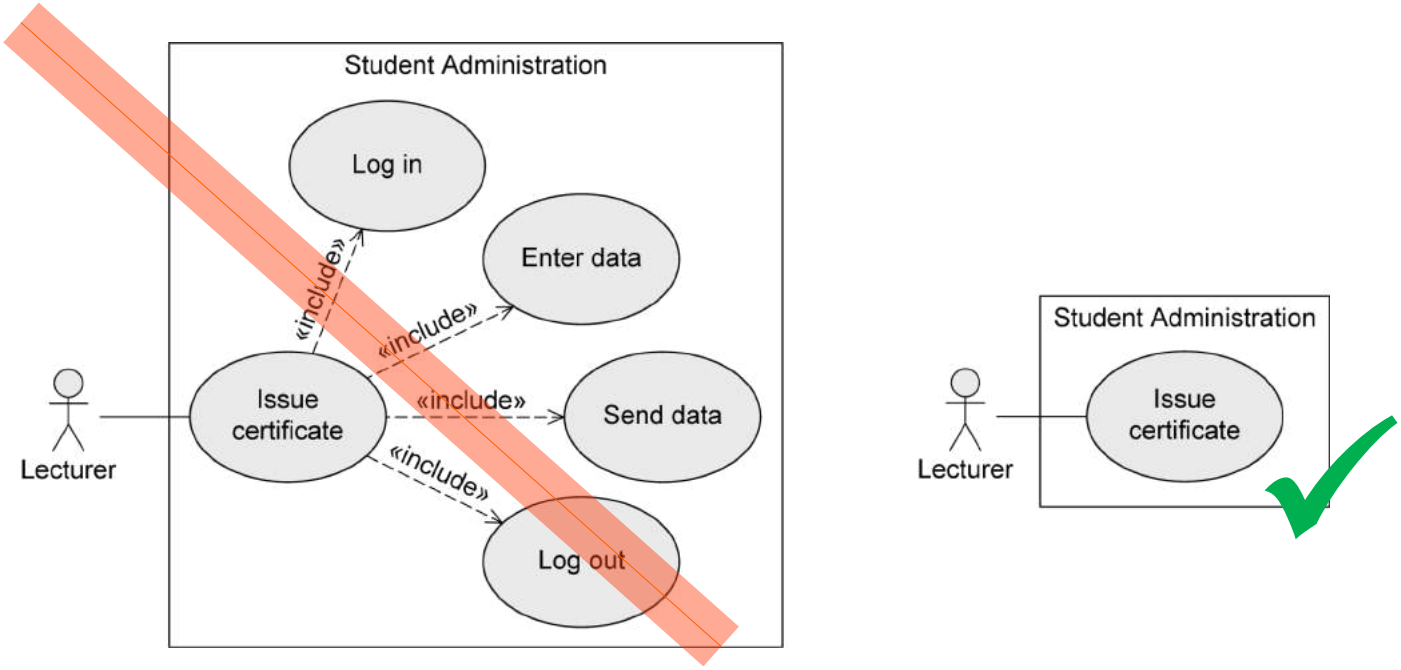
- Beberapa use cases yang kecil yang memiliki tujuan yang sama dapat dikelompokkan menjadi satu use case



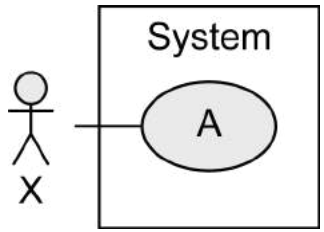
Best Practices

Kesalahan Umum yang Harus Dihindari (5/5)

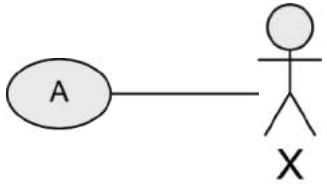
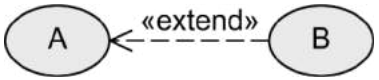
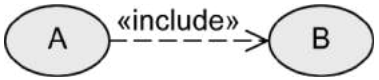
- Langkah-langkah yang harus dilakukan menjadi bagian dari use cases, **bukan** use cases tersendiri! -> BUKAN *functional decomposition*



Notasi Elemen (1/2)

Name	Notation	Description
System		Batas antara sistem dengan pengguna sistem
Use case		Unit fungsionalitas sistem
Actor		Role dari pengguna sistem

Notasi Elemen (2/2)

Name	Notation	Description
Association		Relationship antara use cases dan actors
Generalization		Inheritance relationship antar actors atau use cases
Extend relationship		B extends A: use case B digunakan oleh use case A secara opsional
Include relationship		A includes B: use case B wajib digunakan oleh use case A