PEMODELAN PERANGKAT LUNAK BAG – 1

Telkom University

Budhi Irawan, S.Si, M.T

PEMODELAN DAN UML

- Apa itu *UML*? Yang dimaksud dengan *UML ? UML* merupakan singkatan dari *"*Unified Modelling Language*"* yaitu suatu metode *permodelan* secara *visual* untuk sarana *perancangan sistem berorientasi objek*, atau *definisi UML* yaitu sebagai suatu *bahasa yang sudah menjadi standar pada visualisasi*, *perancangan* dan juga *pendokumentasian sistem software*.
- Saat ini *UML* sudah menjadi *bahasa standar* dalam *penulisan blue print perangkat lunak*.

FUNGSI UML

- Dapat memberikan ***bahasa permodelan visual*** kepada pengguna dari berbagai macam pemerograman maupun proses rekayasa.
- Dapat menyatukan ***praktek-praktek terbaik*** yang ada dalam permodelan.
- Dapat memberikan ***model yang siap untuk digunakan***, merupakan ***bahasa permodelan visual yang ekspresif*** untuk ***mengembangkan sistem*** dan untuk ***saling menukar model secara mudah***.
- Dapat berguna sebagai ***blue print***, sebab ***sangat lengkap dan detail*** dalam perancangannya yang nantinya akan diketahui informasi yang ***detail mengenai koding suatu program***.
- Dapat memodelkan ***sistem yang berkonsep berorientasi objek***, jadi tidak hanya digunakan untuk ***memodelkan perangkat lunak (software) saja***.
- Dapat menciptakan suatu ***bahasa permodelan*** yang nantinya dapat dipergunakan oleh **manusia maupun oleh mesin**.

DIAGRAM UML (1)

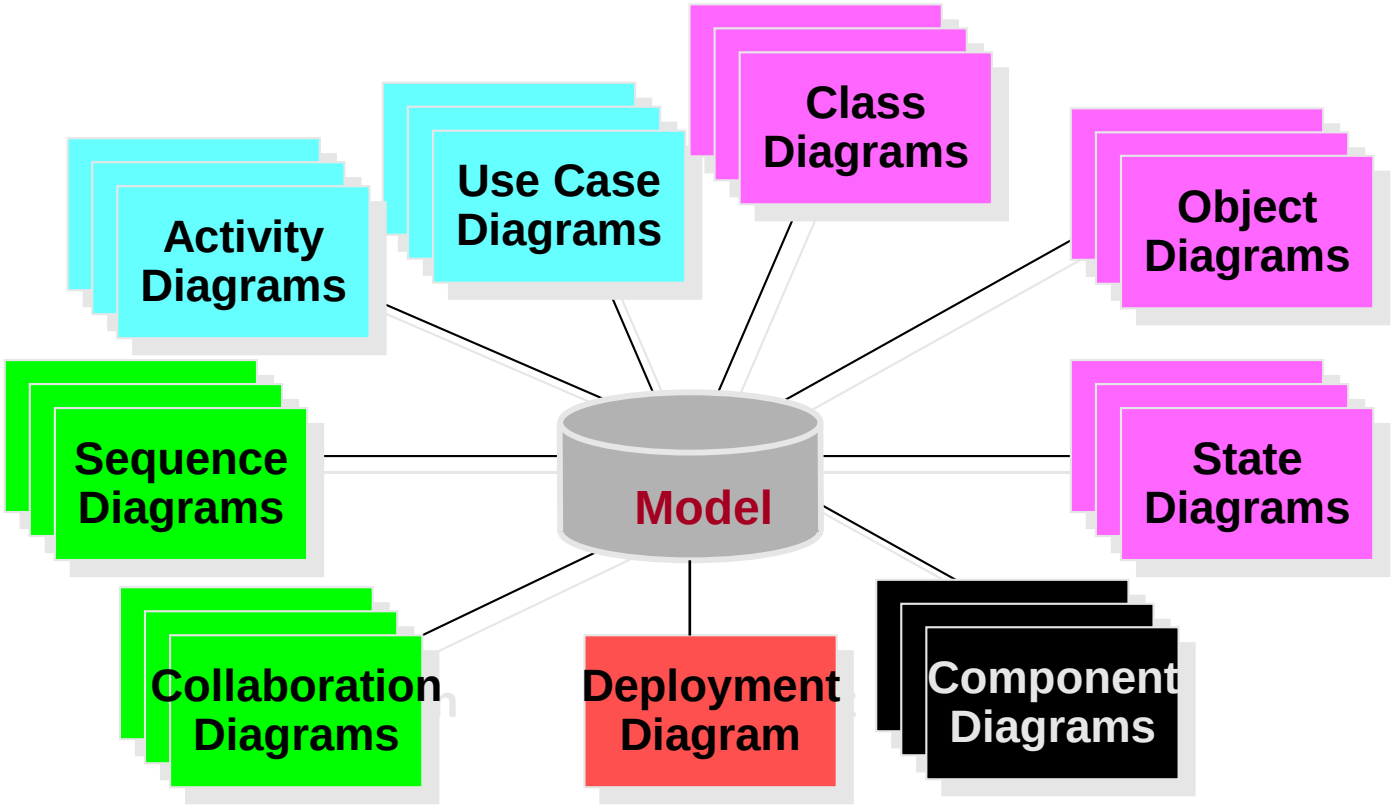


DIAGRAM UML (2)

1. Class Diagram
2. Object Diagram
3. Component Diagram
4. Deployment Diagram

Structural Diagrams

5. Use Case Diagram
6. Sequence Diagram
7. Collaboration Diagram
8. Statechart Diagram
9. Activity Diagram

Behavioral Diagrams

DIAGRAM UML (3)

Diagram	Kegunaan
Use Case Diagram	Menunjukan aktor (orang atau pengguna sistem), use case (skenario ketika mereka menggunakan sistem) dan beserta relasinya
Activity Diagram	Menunjukan aktivitas dan perubahan dari satu aktivitas ke aktivitas lainnya dengan kondisi yang terjadi pada beberapa bagian dari sistem
Sequence Diagram	Menunjukan objek – objek dan suatu urutan untuk memanggilnya serta membuat objek lainnya
Class Diagram	Menunjukan class dan relasi diantaranya
Collaboration Diagram	Menunjukan objek dan hubungannya, menempatkan penekanan pada objek yang berpartisipasi dalam kerangka pertukaran pesan
State Diagram	Menunjukan state, perubahan state dan kejadian dalam objek atau bagian dari sistem
Deployment Diagram	Menunjukan contoh dari komponen dan relasinya
Component Diagram	Menunjukan level tertinggi komponen pemrograman
Entity Relationship Diagram	Menunjukan data dan relasinya serta batasan diantara data

USE CASE DIAGRAM

- Menggambarkan ***fungsionalitas*** yang diharapkan dari sebuah sistem. Yang ditekankan adalah "***apa***" yang diperbuat sistem, dan bukan "***bagaimana***".
- Menggambarkan ***kebutuhan sistem dari sudut pandang user***
- Mengfokuskan pada ***proses komputerisasi (automated processes)***
- Menggambarkan ***hubungan antara use case dan actor***
- Use case menggambarkan p***roses sistem (kebutuhan sistem dari sudut pandang user)***

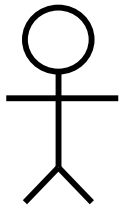
ELEMEN USE CASE DIAGRAM

1. Use case
   - Menjabarkan ***aktifitas aktor*** dalam ***sistem*** yang memberikan ***hasil yang bisa dilihat***
2. Aktor
   - ***Entitas eksternal (di luar sistem)*** yang ***berinteraksi*** dengan ***sistem yang berpartisipasi dalam use case***
   - Bisa berupa ***orang, pengguna***, ***sistem lain***, atau ***event eksternal***
3. Deskripsi Use Case
   - ***Teks narasi*** dari Use Case ***berupa catatan atau dokumen*** yang terhubung ke Use Case dan ***menjelaskan proses atau aktivitas yang dilakukan di Use case***
4. Boundary Sistem

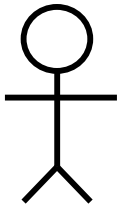
ACTOR (1)

- *Actor* menggambarkan *orang, system atau external entitas / stakeholder* yang *menyediakan atau menerima informasi dari system*
- *Actor* menggambarkan *sebuah tugas/peran dan bukannya posisi sebuah jabatan*
- *Actor* memberi *input* atau menerima *informasi dari system*
- Actor biasanya *menggunakan Kata benda*
- *Tidak boleh ada komunikasi langsung antar actor*
- Letakkan *actor* utama anda pada pojok *kiri atas dari diagram*

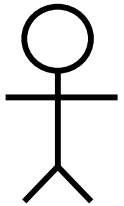
ACTOR (2)



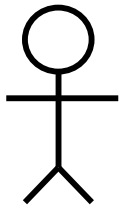
Manajer Proyek



Manajer Sumber Daya



Sumber Daya Manusia



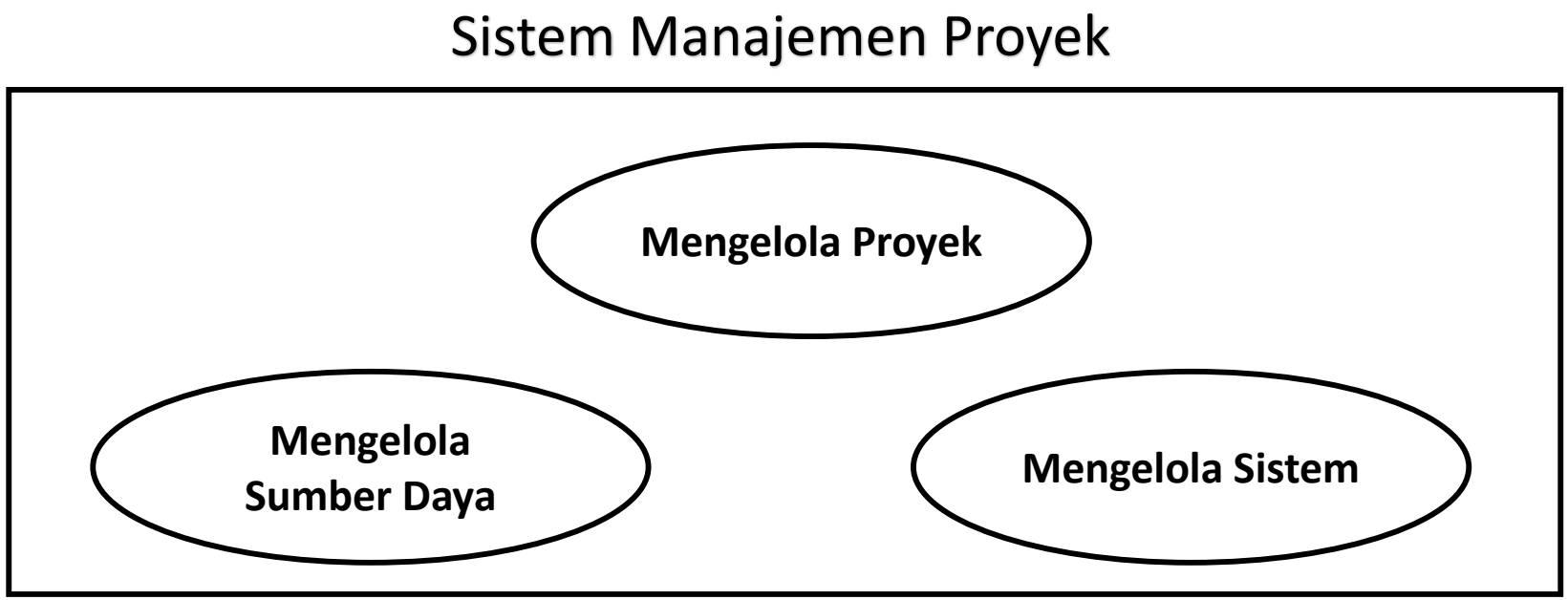
Administrator Sistem

<<actor>>
Backup System

USE CASE (1)

- Merupakan ***deskripsi interaksi*** antara ***user/pengguna dengan sistem***
  - Merepresentasikan ***antarmuka eksternal*** dari sistem
  - Menspesifikasikan ***requirement tentang apa*** yang sistem harus lakukan (bukan bagaimana)
- Aturan use case:
  - Biasanya menggunakan ***kata kerja (verb)***
  - Tiap use case ***mempunyai relasi dengan setidaknya satu aktor***
  - Tiap use case ***mempunyai seorang inisiator, yaitu seorang aktor***
  - Tiap use case ***mengarah ke hasil yang relevan dengan "nilai bisnis"***

USE CASE (2)

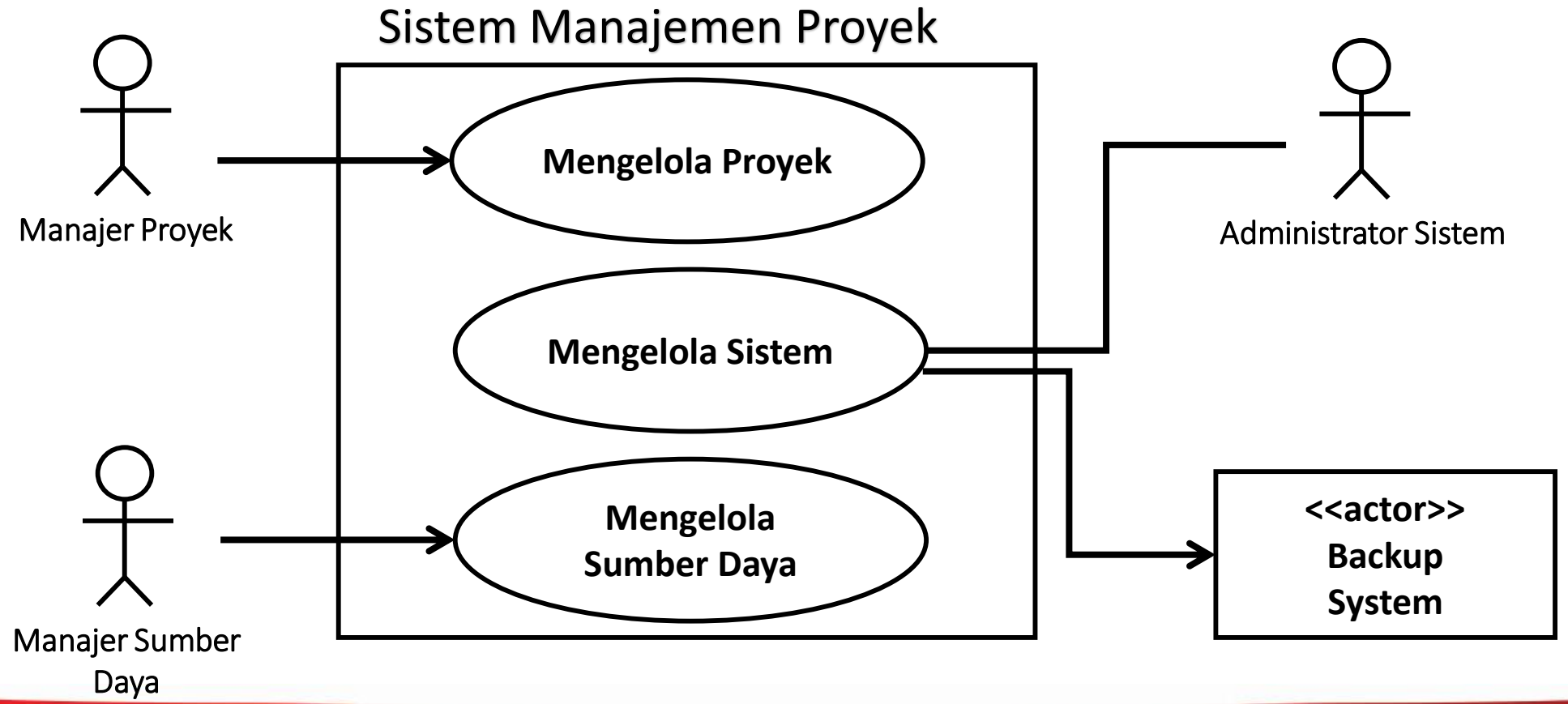

Sistem Manajemen Proyek
Mengelola Proyek
Mengelola Sumber Daya
Mengelola Sistem

RELASI USE CASE (1)

- ***Relasi Use Case*** dengan ***Actor*** disebut ***Asosiasi***
  - Ujung panah pada association antara ***actor*** dan ***use case*** mengindikasikan ***siapa/apa*** yang ***meminta interaksi*** dan bukannya mengindikasikan aliran data
  - Sebaiknya gunakan **Garis tanpa panah** untuk ***asosiasi*** antara ***actor*** dan ***use case***

  - ***asosiasi*** antara ***actor*** dan ***use case*** yang menggunakan **panah terbuka** untuk ***mengindikasikan*** bila ***actor*** berinteraksi secara ***pasif*** dengan system anda

RELASI USE CASE (2)



RELASI USE CASE (3)

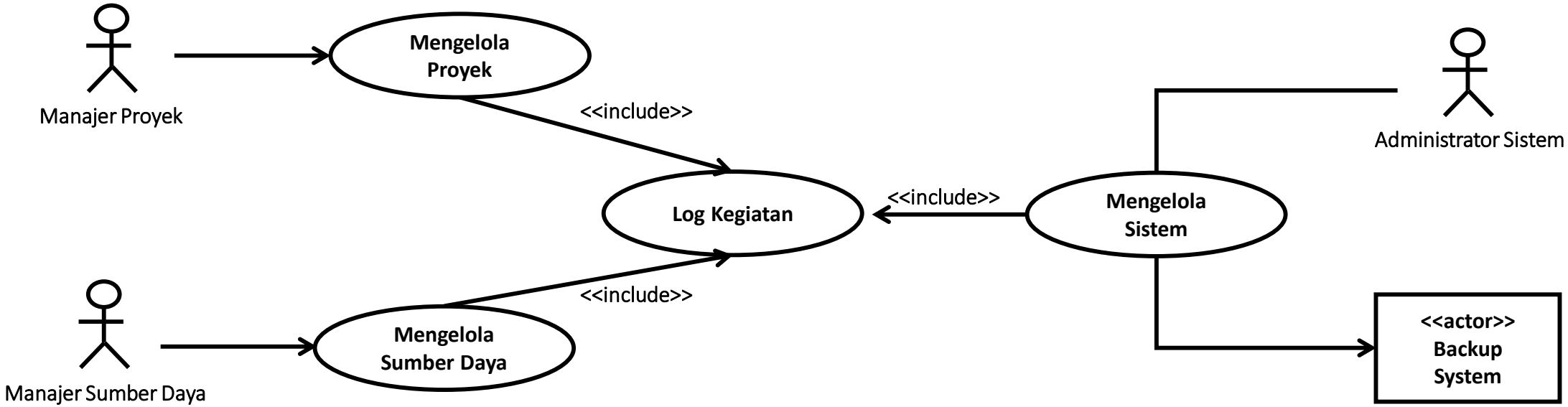
- *Use Case* dapat mempunyai ***relasi*** dengan ***Use Case lain***
  1. include
     - Menspesifikasikan ***sebuah Use Case*** yang ***berada di dalam Use Case lain***
  2. extends
     - Menspesifikasikan bahwa dalam situasi tertentu (disebut sebagai titik ekstensi), ***sebuah Use Case akan diperluas oleh yang lain***
  3. Generalisasi (*Use case Generalization*)
     - Menspesifikasikan sebuah ***Use Case yang mewarisi karakteristik*** dari ***Use Case 'Super'***, menggunakan kembali ***perilaku yang sama untuk beberapa Use Case***

1. RELASI INCLUDE (1)

- <<include>> : *termasuk didalam use case lain (required) / (diharuskan)*
  - ***Pemanggilan use case oleh use case lain***, contohnya adalah pemanggilan sebuah fungsi program
  - ***Tanda panah terbuka*** harus ***terarah ke sub use case***
  - Gambarkan ***asosiasi include*** secara sebaiknya ***horizontal***

1. RELASI INCLUDE (2)

- Mirip dengan pemanggilan fungsi atau sub-rutin

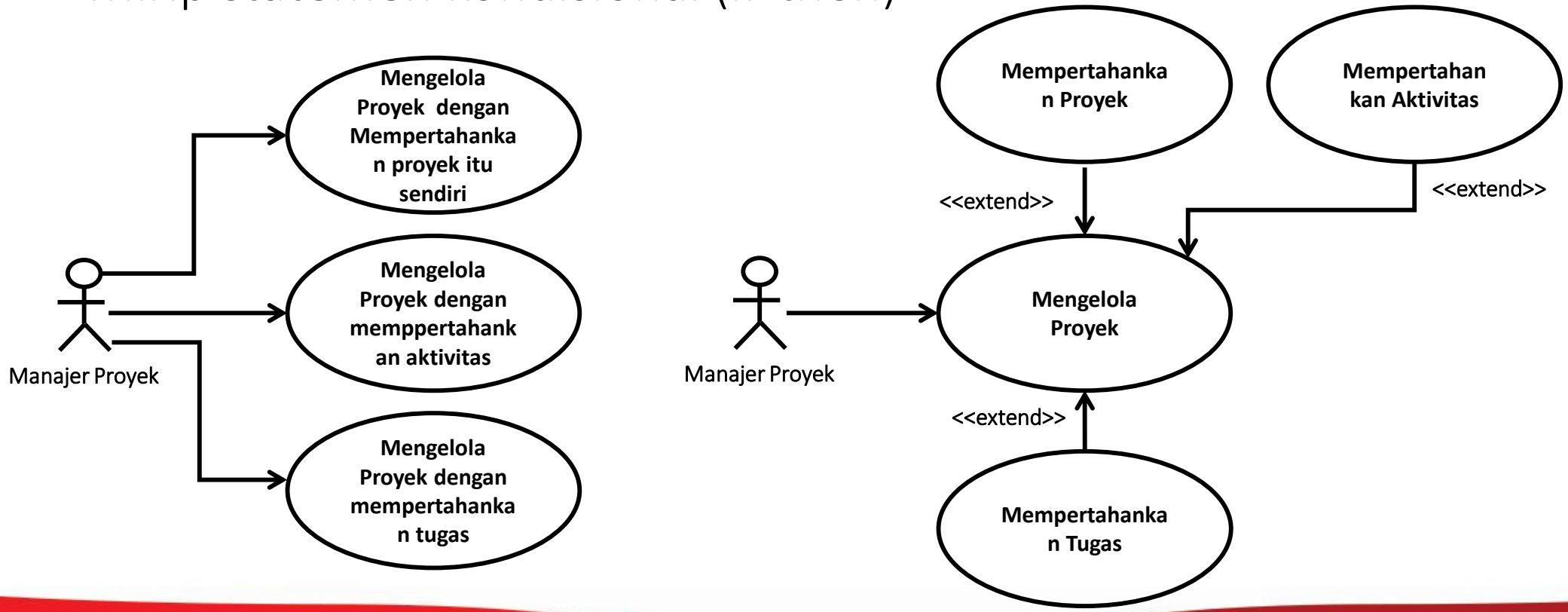


2. RELASI EXTEND (1)

- <<extend>> : *perluasan dari use case lain jika kondisi atau syarat terpenuhi*
  - *Kurangi penggunaan asosiasi Extend* ini, terlalu banyak pemakaian asosiasi ini membuat diagram ***sulit dipahami***.
  - Tanda panah terbuka harus terarah ke parent/base use case
  - Gambarkan association ***extend*** sebaiknya secara vertical

2. RELASI EXTEND (2)

- Mirip statemen kondisional (if-then)



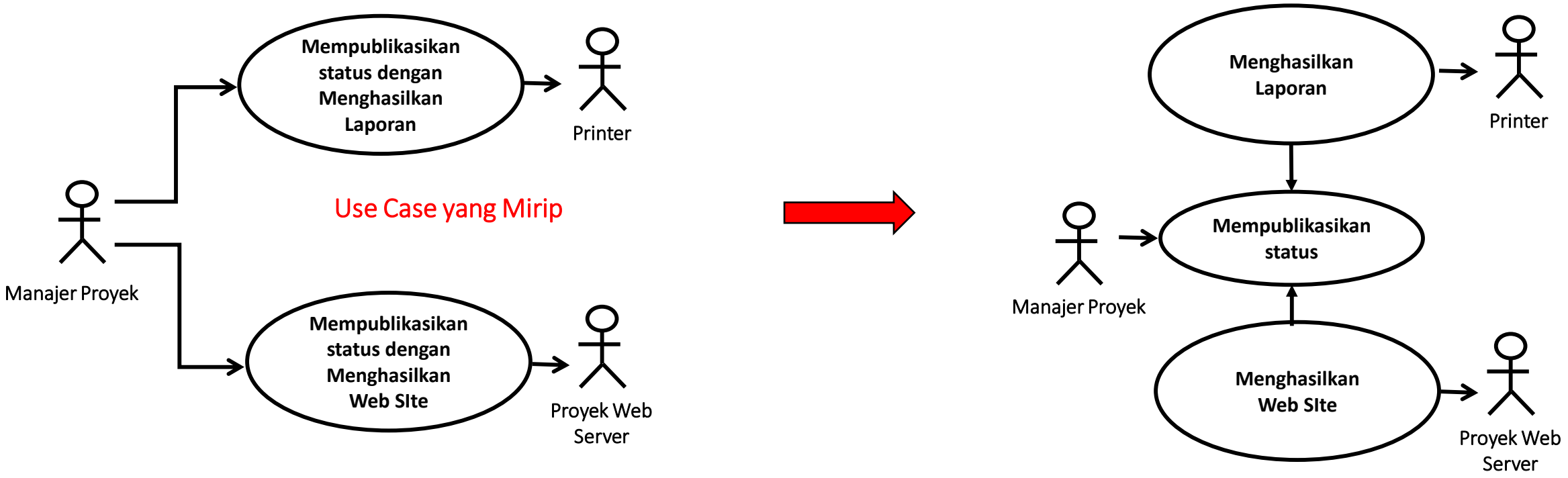
3. RELASI GENERALISASI (1)

- *Generalization/inheritance* digambarkan dengan sebuah *garis berpanah tertutup* pada salah satu ujungnya yang menunjukkan lebih umum

- Gambarkan *generalization/inheritance* antara use case sebaiknya *secara vertical* dengan *inheriting use case dibawah base/parent use case*
- *Generalization/inheritance* dipakai ketika ada sebuah keadaan yang lain *sendiri/perlakuan khusus (single condition)*

3. RELASI GENERALISASI (2)

- Dua buah Use Case dengan perilaku mirip digeneralisasikan



CONTOH USE CASE DIAGRAM

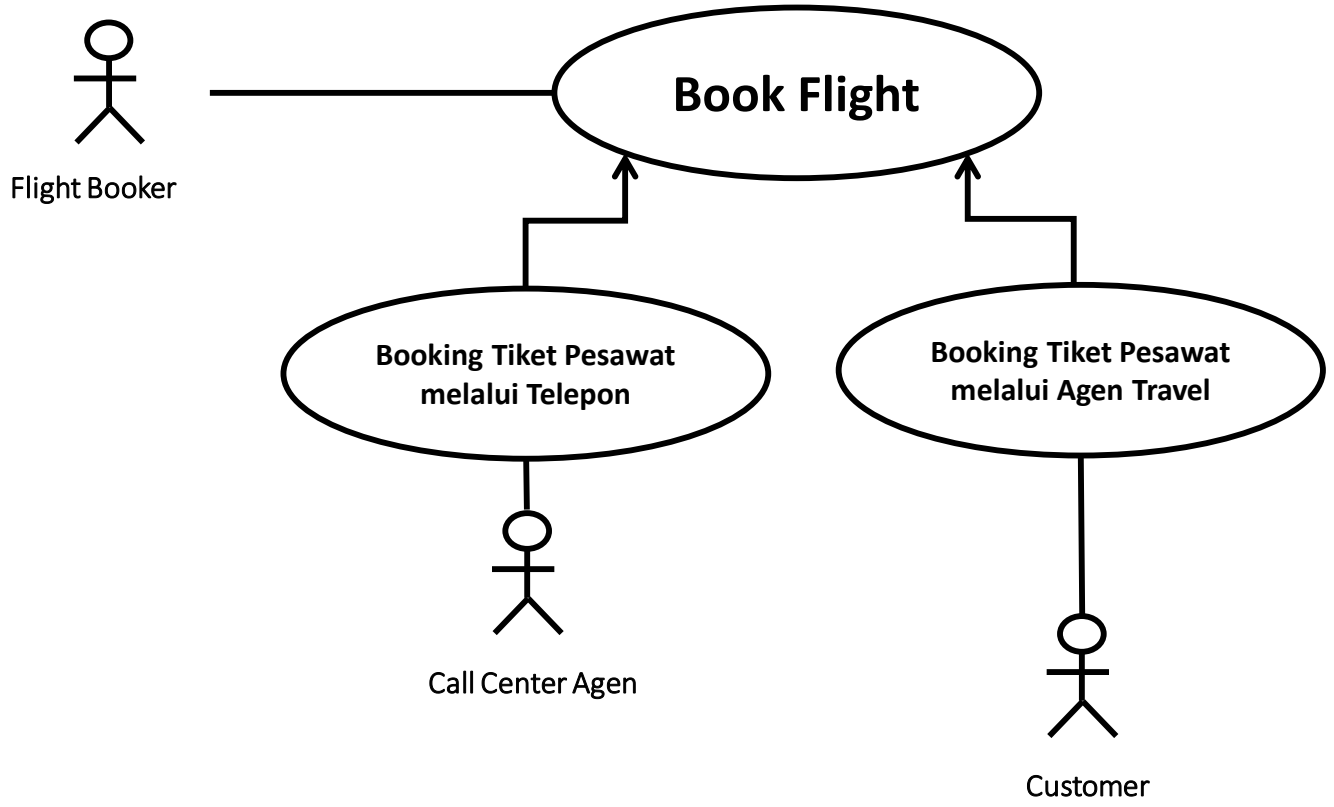
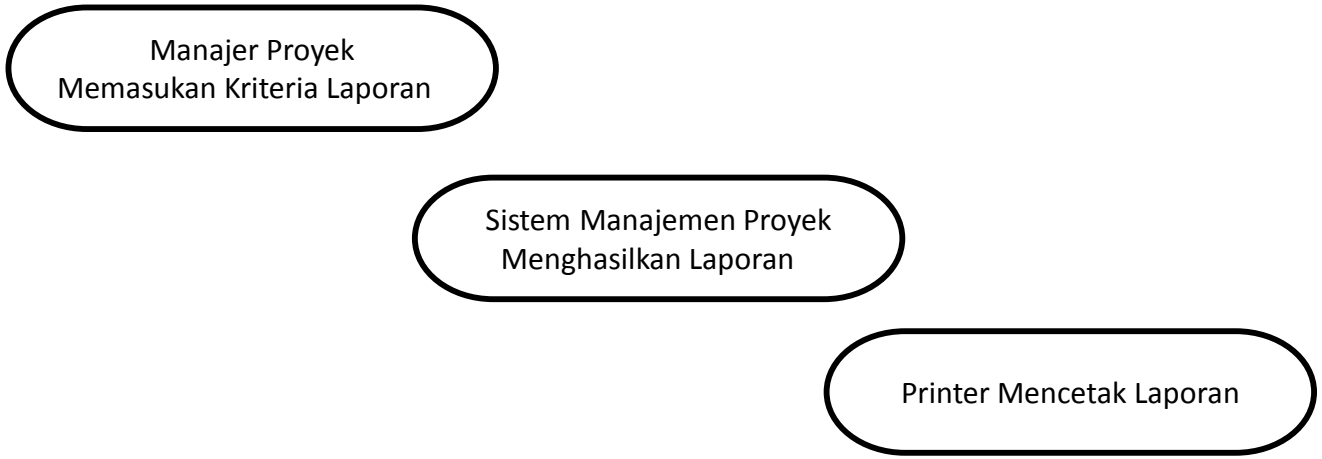


DIAGRAM AKTIVITAS

- Menjelaskan tentang ***urutan aktivitas dalam sistem***
  - Pemodelan perilaku sistem
- ***Diagram aktivitas*** selalu ***terasosiasi ke sebuah Class***, ***sebuah Operator*** dan ***sebuah Use Case***
- Diagram ini bisa ***aktivitas sekuensial (berurut)*** dan ***paralel***
  - ***Paralel*** dilakukan dengan ***fork/wait***
  - Urutan aktivitas dalam ***eksekusi paralel*** tidak dipentingkan (***bisa dilakukan di waktu yang sama atau tidak***)

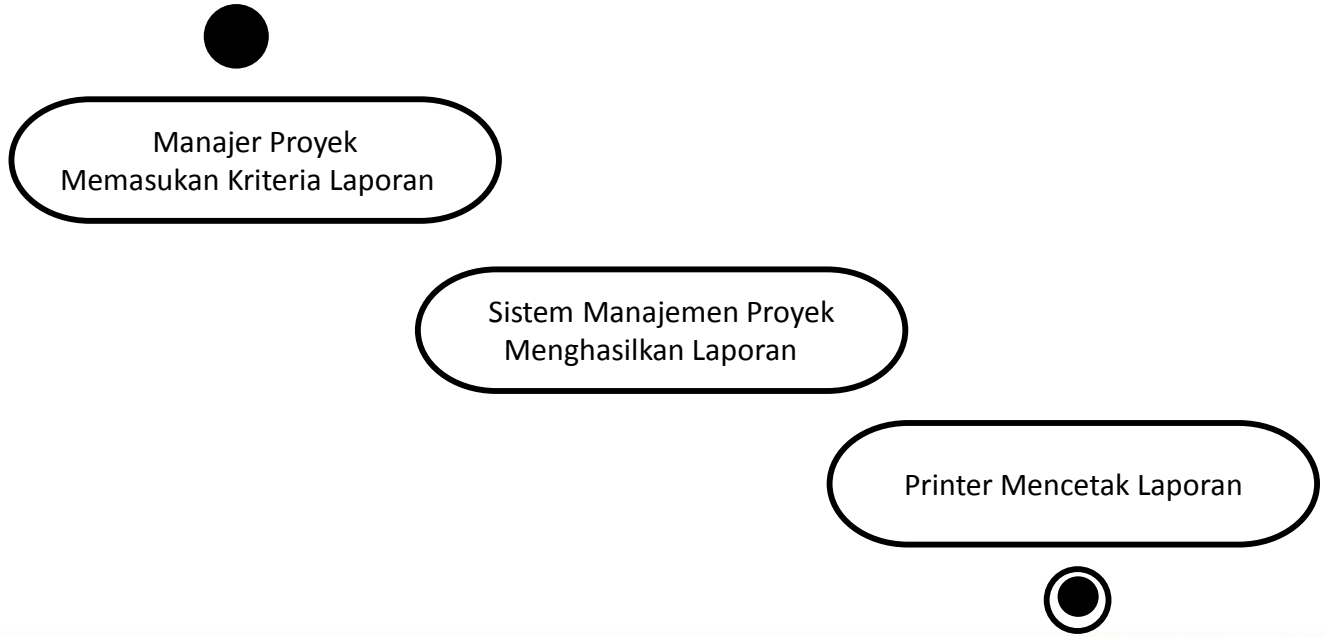
ACTION STATE

- ***Action state*** merepresentasikan sebuah ***proses*** yang dilakukan oleh sebuah ***elemen***. Misalnya:



ACTION STATE INISIAL DAN FINAL

- *Action state inisial* : *action pertama* yang dijalankan dalam diagram aktivitas
- *Action state final* : *action terakhir* yang dijalankan dalam diagram aktivitas

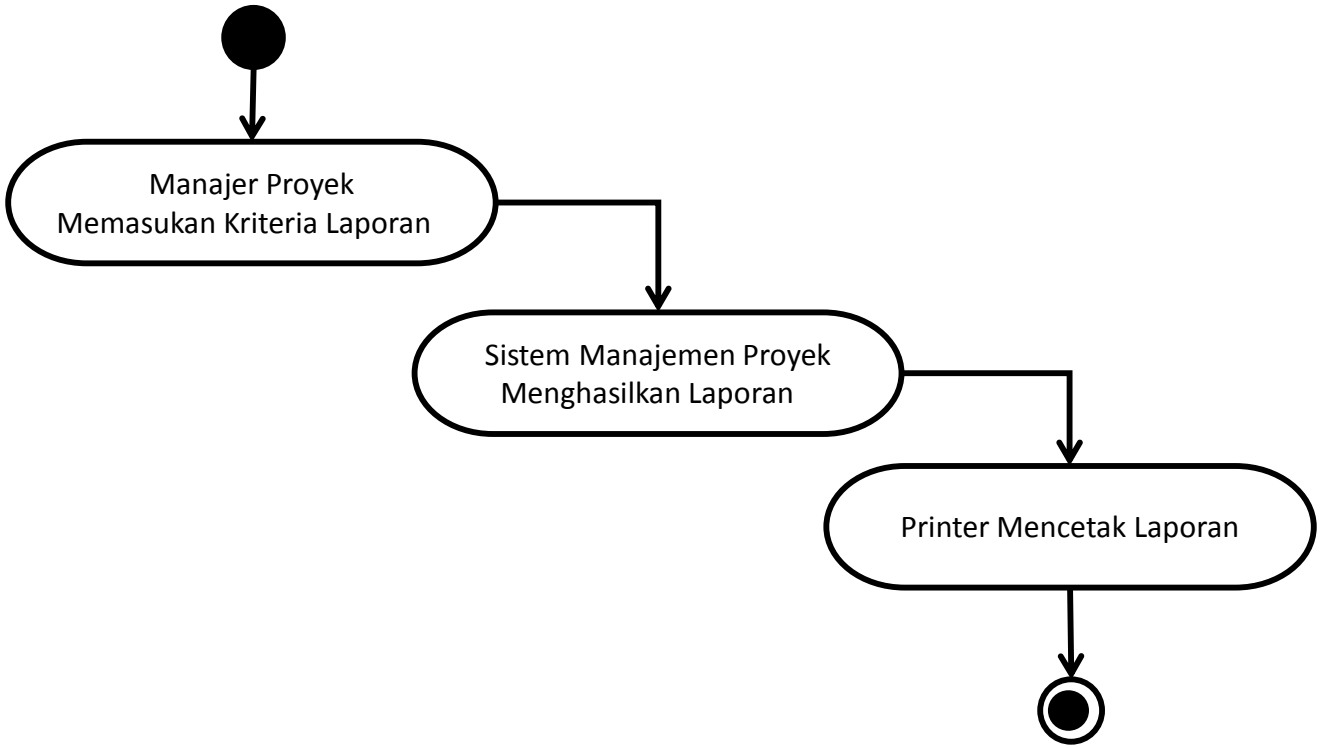


TRANSISI ALIRAN

- Tipe transisi:
  1. *Control flow* atau *default transition* atau *automatic transition*
     - Karena *tidak mempunyai label* dan *seketika ditrigger setelah sumber state action selesai diproses (Menunjukkan urutan eksekusi atau action state).*
  2. *Object flow*
     - Menunjukkan *aliran objek* dari *sebuah action atau activity* ke *action atau activity lain*.

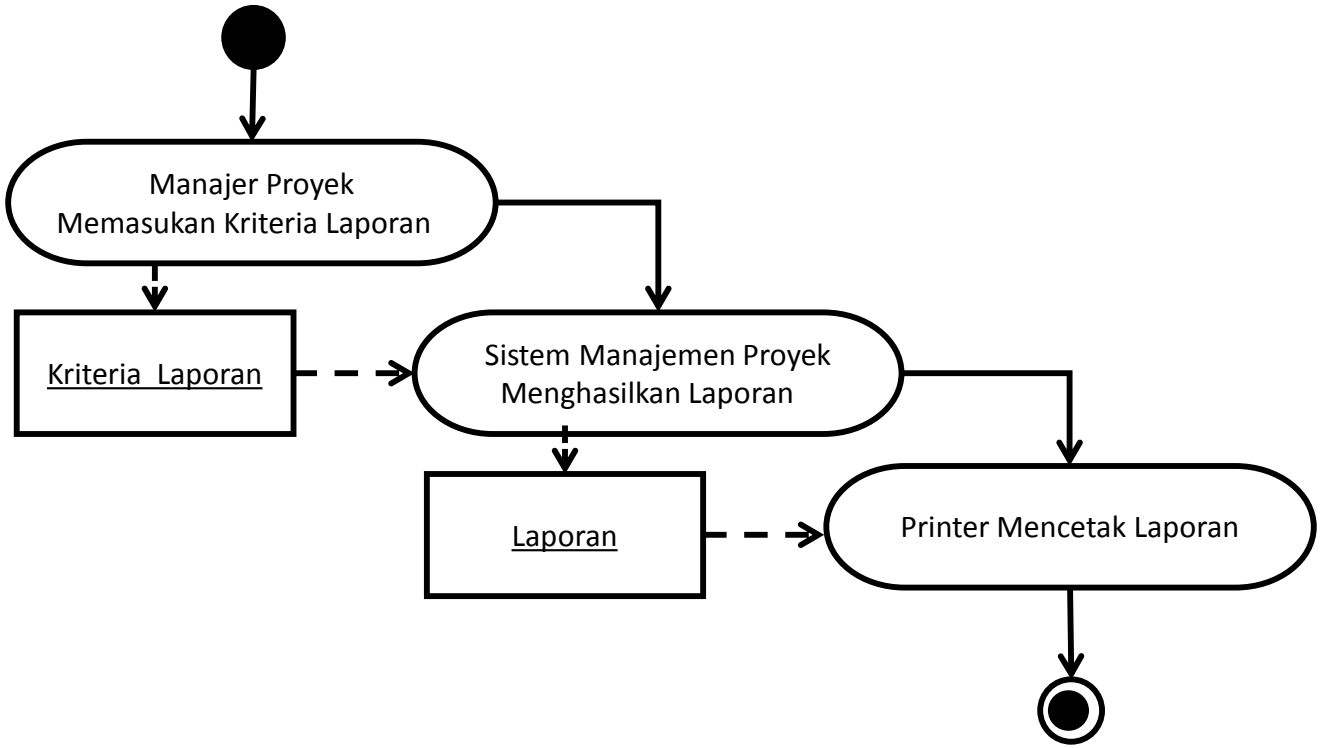
1. CONTROL FLOW

- Menunjukkan *urutan dari action state*
  - Jika *action state sumber telah terproses, action state target dapat mulai diproses*
  - Ditunjukkan dengan **garis tebal**



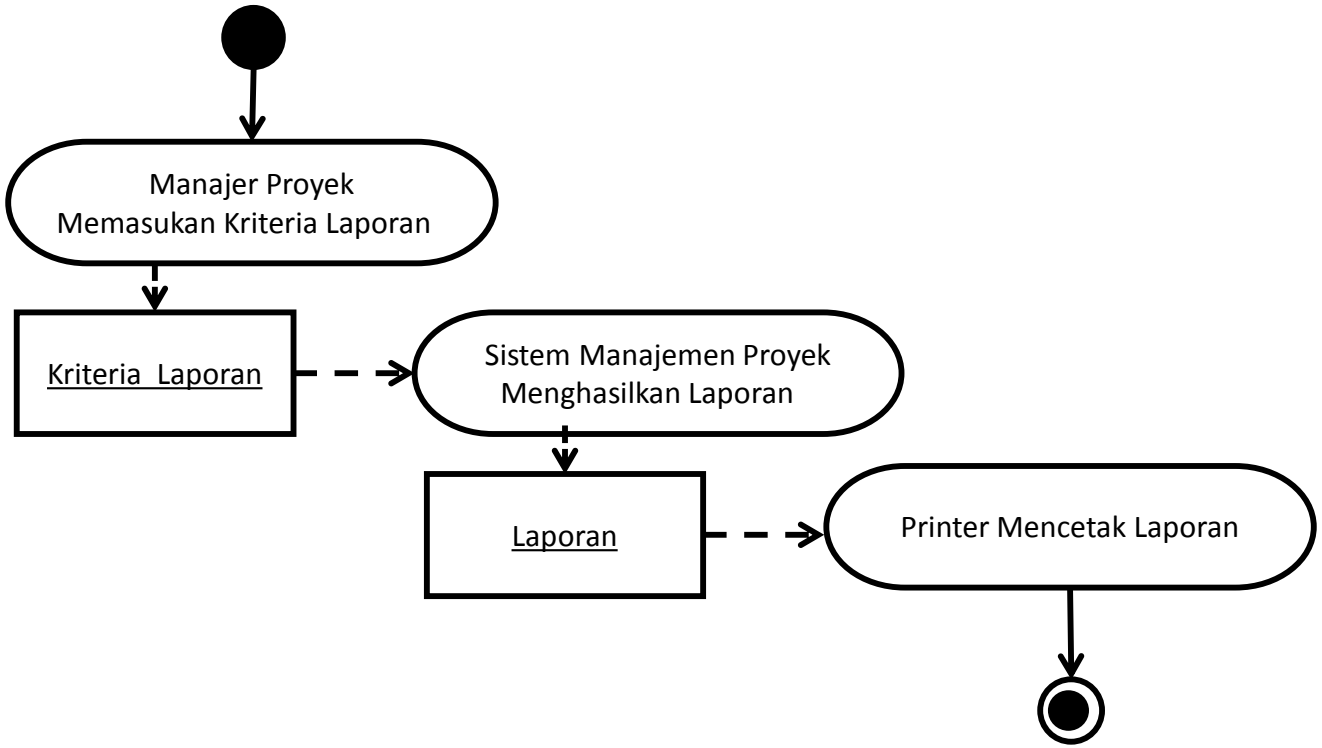
2. OBJECT FLOW (1)

- Menunjukkan bahwa ***sebuah action state memasukkan*** atau ***menghasilkan sebuah object***. Misalnya:
  - ***Action state*** : ***"Manajer Proyek memasukan Kriteria Laporan"*** Menghasilkan object Kriteria Laporan



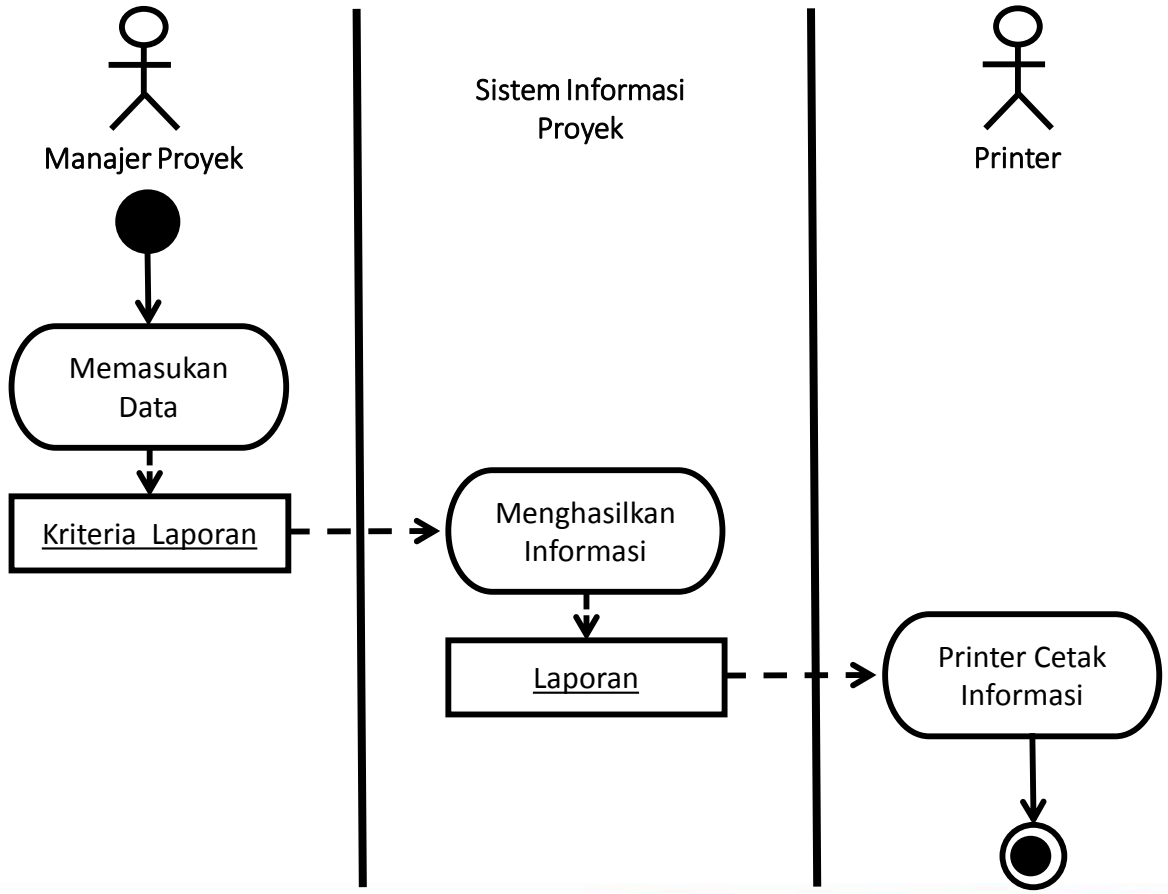
2. OBJECT FLOW (2)

- Diagram sebelumnya *object input* dan *output* telah dideklarasikan secara eksplisit, sehingga *control flow bisa dihilangkan*



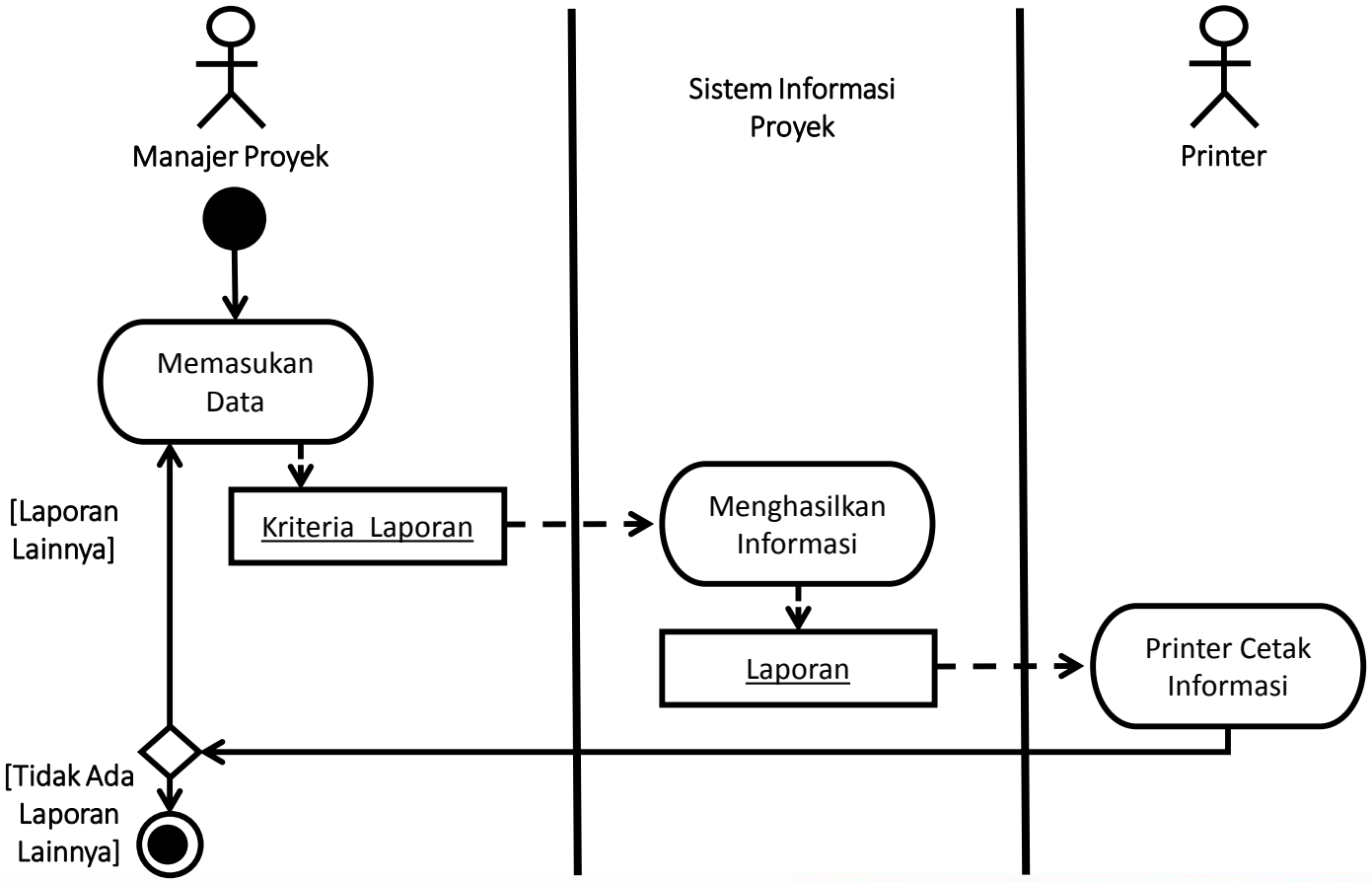
SLIMLANE

- Merupakan ***daerah visual*** dalam ***diagram aktivitas*** yang menghindikasikan ***elemen yang bertanggung jawab terhadap action state*** dalam daerah tersebut



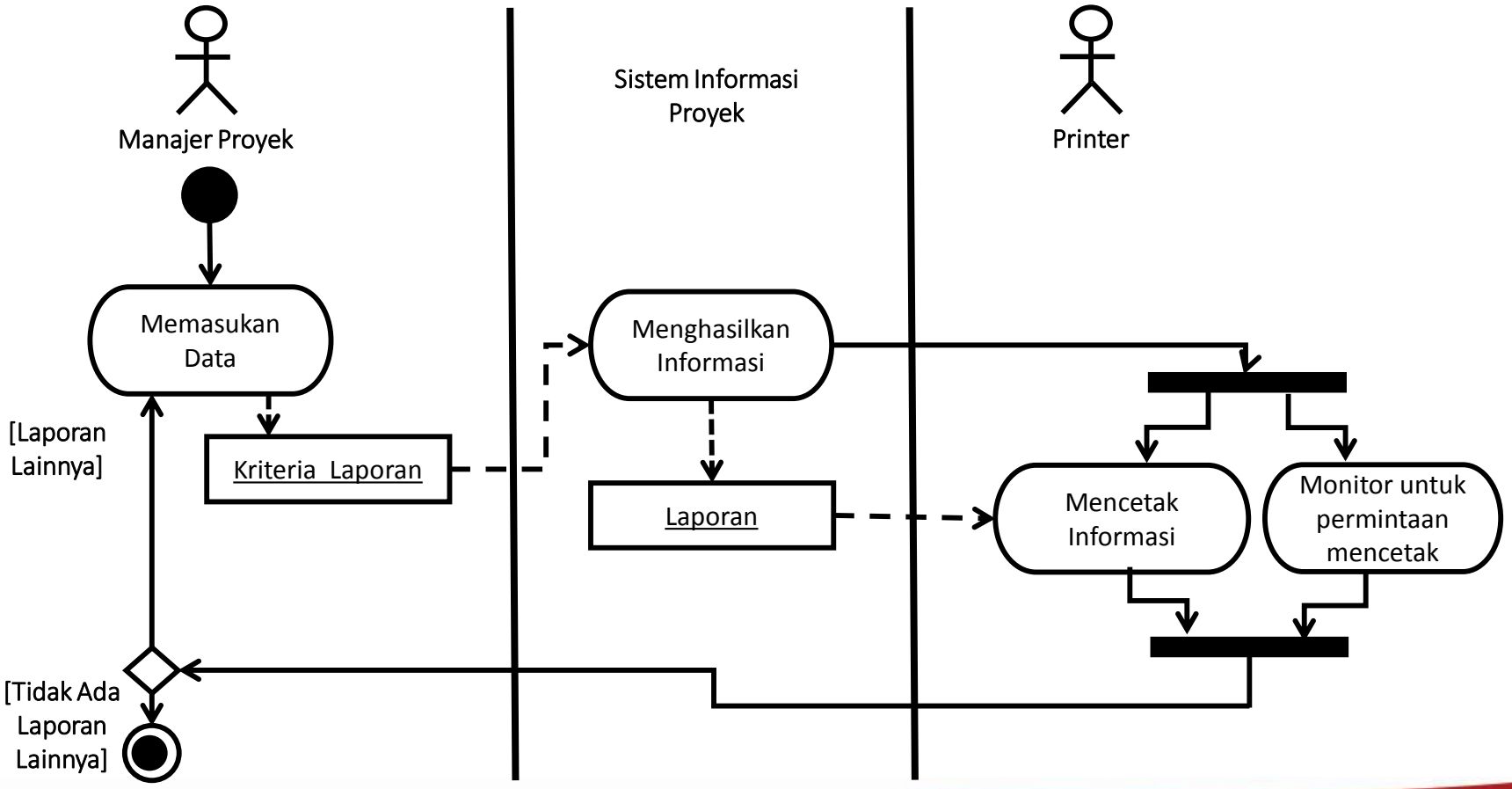
PENGAMBILAN KEPUTUSAN

- *Pengambilan keputusan* dilakukan dengan *memilih salah satu control-flow* sesuai *dengan kondisi yang diinginkan*



CONCURRENCY

- *Concurrency* memilih beberapa *transisi sekaligus*
  - Proses : *pemecahan kontrol* dan *sinkronisasi kontrol*



CONTOH DIAGRAM AKTIVITAS

Proses Bussiness : Make Reservation