Lanjutan Chapter 8
Pemodelan Analisis

R.S. Pressman & Associates, Inc.
A. Haidar Mirza, M.Kom
Siti Sa'uda, M.Kom.

Pemodelan berorientasi aliran

Menampilkan bagaimana objek data ditransformasi ketika mereka bergerak di dalam sistem

Sebuah data flow diagram (DFD) merupakan bentuk diagram yang digunakan

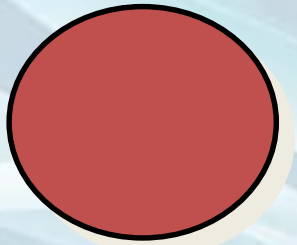
Walaupun dianggap pendekatan kuno, pemodelan berorientasi aliran menyediakan pandangan unik terhadap suatu sistem. Dia tetap layak digunakan untuk mendukung analisis elemen model lainnya.

Model Aliran

Setiap sistem berbasis komputer
Adalah sebuah transformasi informasi

Notasi Model Aliran

Entitas Eksternal



proses

Aliran data

Penyimpanan data

Entitas Eksternal

Produsen atau konsumen sebuah data

Contoh : seseorang, piranti, sensor

Contoh lain : sistem berbasis komputer

Data harus selalu berawal dari suatu tempat dan
Harus selalu dikirim pada sesuatu

Proses

Sebuah transformer data
(mengubah input menjadi output)

Contoh: menghitung pajak, menentukan luas,
Memformat laporan, menampilkan grafik

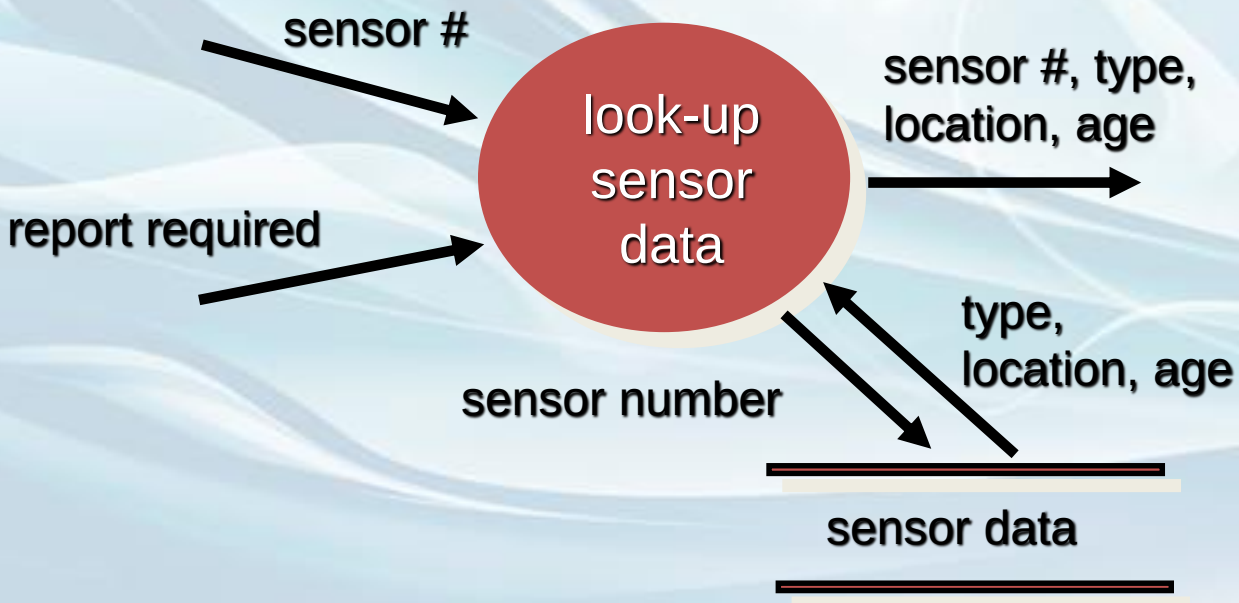
Data harus selalu diproses dalam bentuk tertentu
Untuk menerima fungsi sistem

Aliran Data

Data mengalir melalui sebuah sistem dimulai Sebagai input dan ditransformasi menjadi output

Menyimpan Data

Data disimpan untuk digunakan lagi.



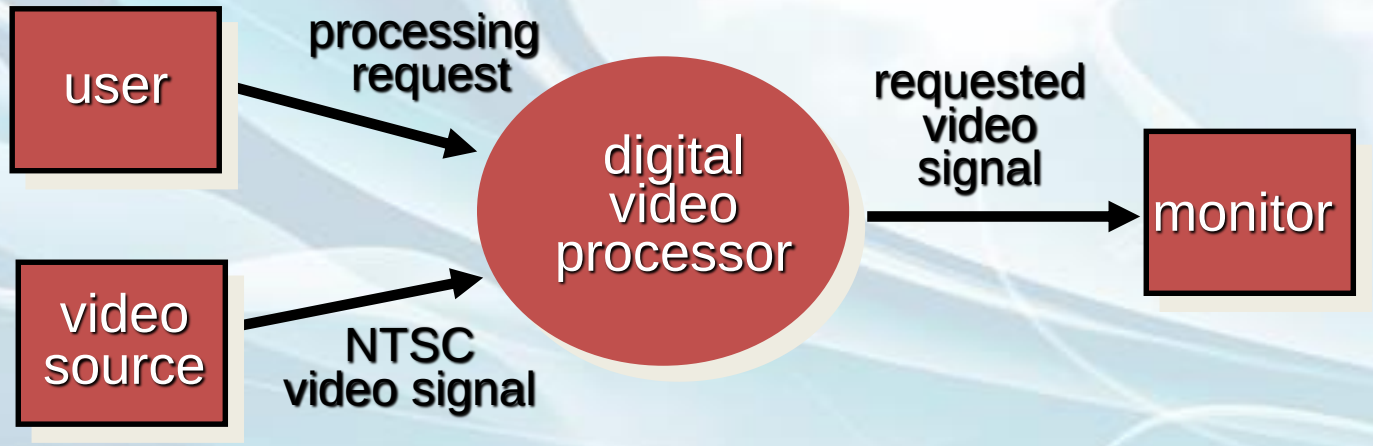
Petunjuk menggambar DFD

- Semua icon harus diberi nama yang bermakna jelas
- DFD berkembang dalam beberapa tingkatan
- Selalu dimulai dengan sebuah context level diagram (level 0)
- Selalu menunjukkan entitas eksternal pada level 0
- Selalu berinama panah aliran data
- Jangan menampilkan prosedur logika

Membangun sebuah DFD—I

- Mereview model data untuk mengisolasi objek data dan gunakan parsing gramatikal untuk menentukan “Operasi”
- Menentukan entitas eksternal (produsen dan konsumen data)
- Membuat level 0 DFD

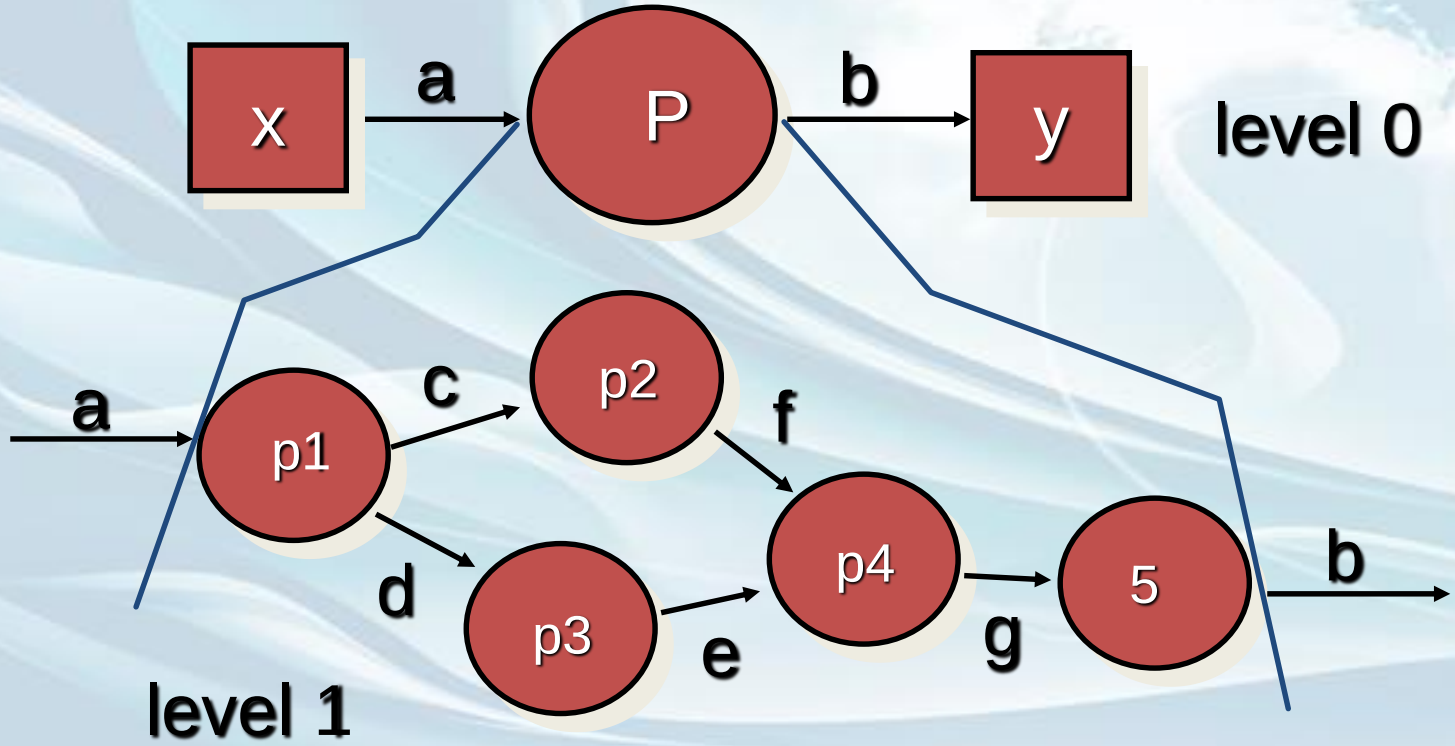
Contoh Level 0 DFD



Membangun DFD—II

- Tulis sebuah narasi yang menggambarkan transformasi
- Parsing untuk menentukan transformasi tingkat berikutnya
- “seimbangkan” aliran untuk menjaga aliran data
- Bangun level 1 DFD
- Gunakan rasio 1:5 (perkiraan)

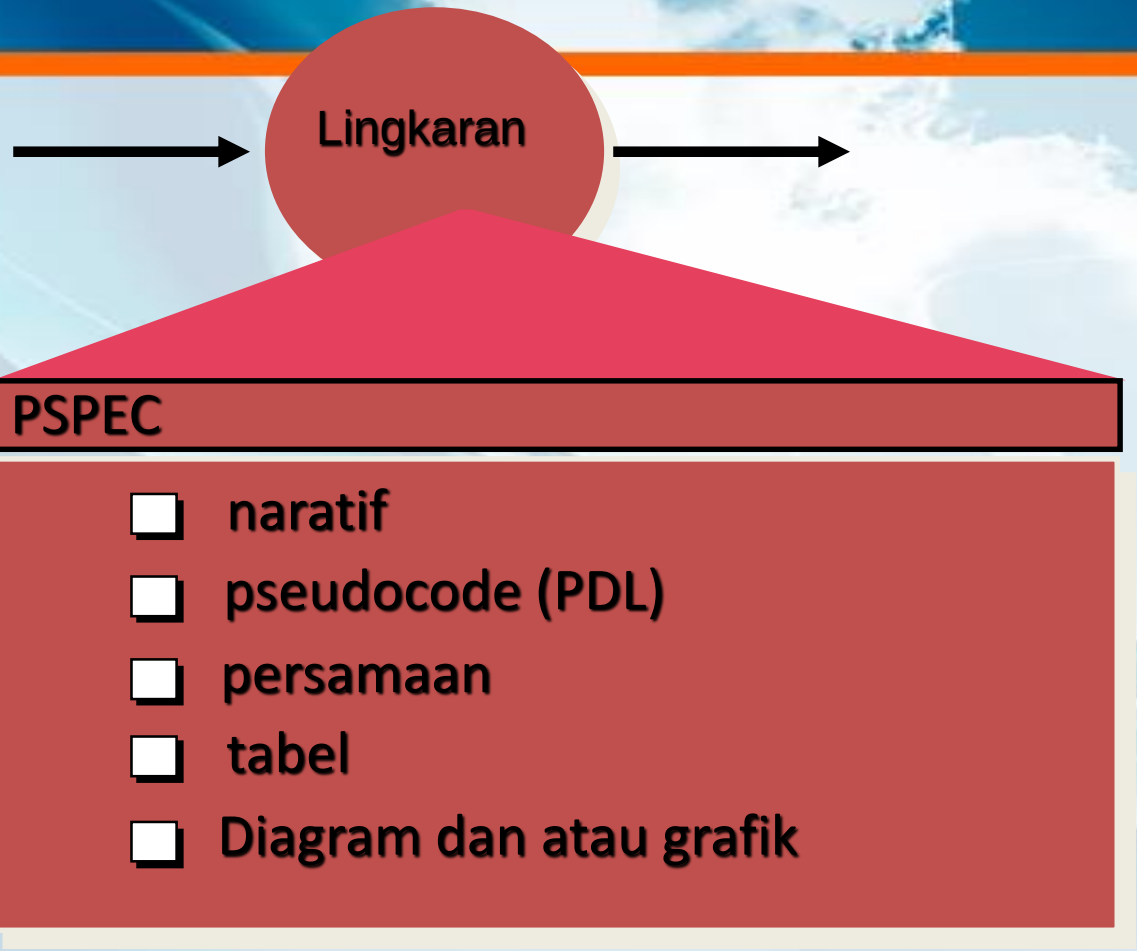
Hierarki Aliran Datang



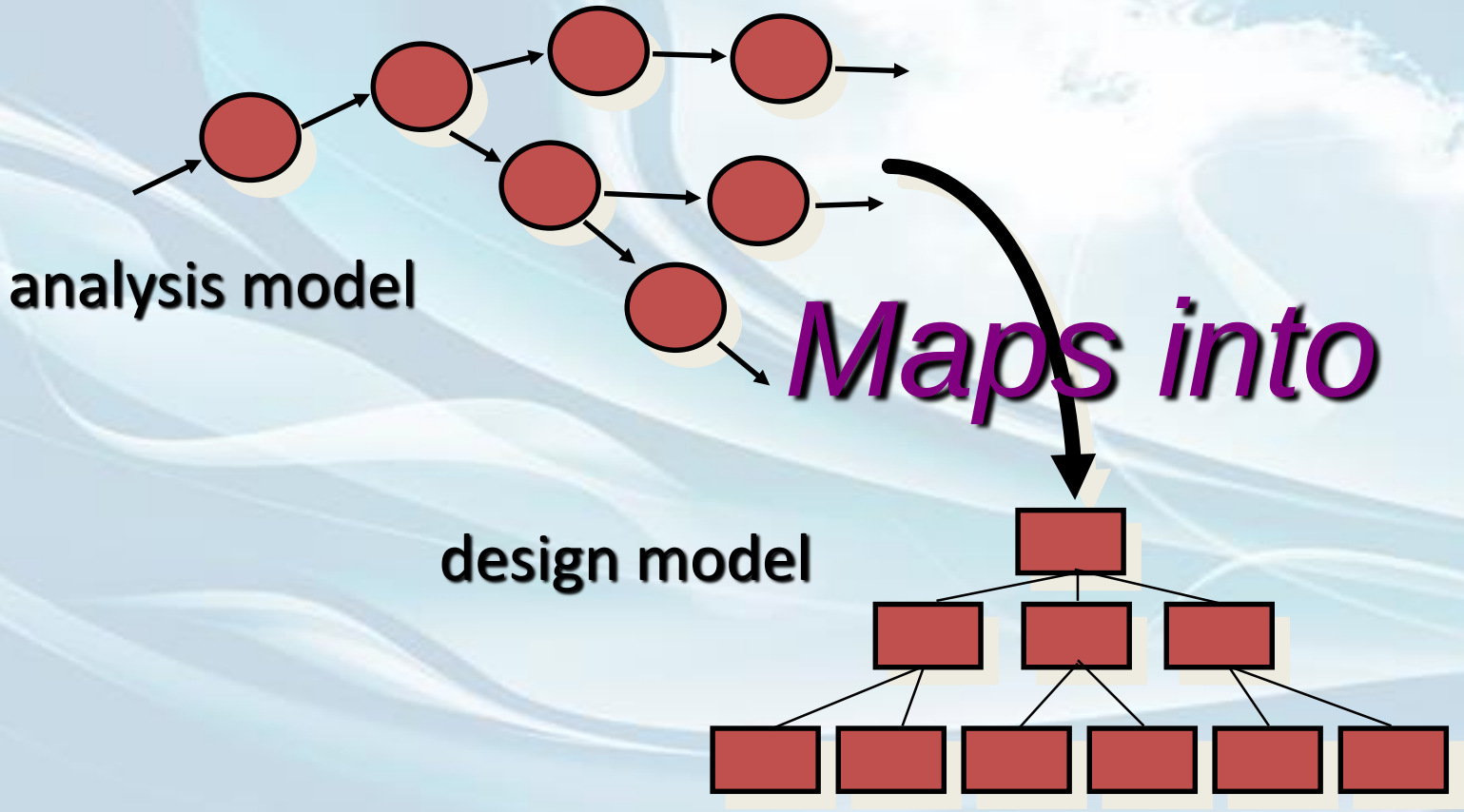
Catatan untuk DFD

- Setiap lingkaran harus dipecah hingga dia hanya melakukan hanya SATU hal
- Rasio ekspansi menurun sesuai dengan jumlah level yang meningkat
- Kebanyakan sistem membutuhkan antara 3 hingga level 7 untuk model aliran yang cukup.
- Sebuah item aliran data (panah) dapat dikembangkan seiring dengan meningkatnya level (data dictionary menyediakan informasi ini)

Spesifikasi Proses (PSPEC)



Setelah DFD?



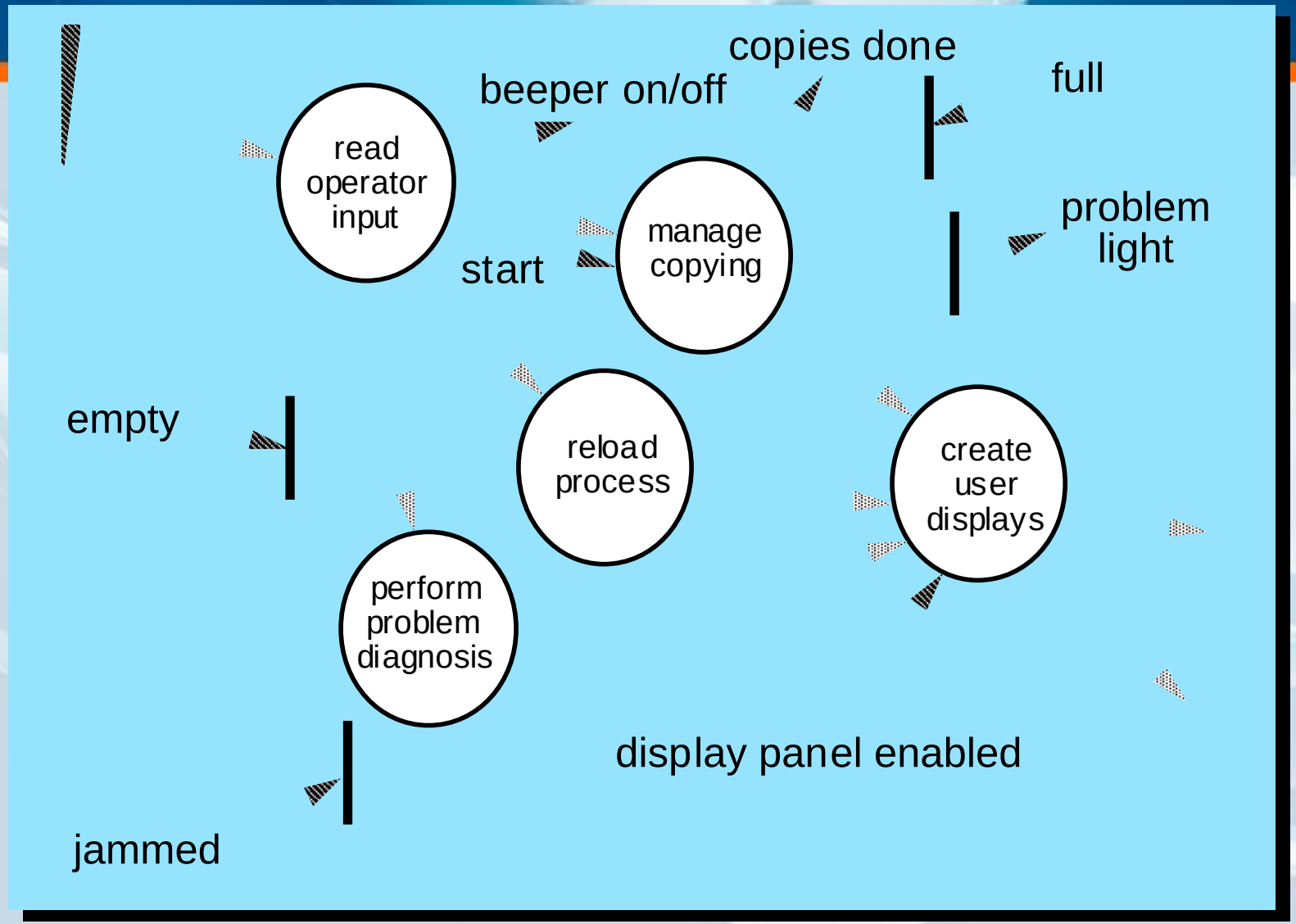
Control Flow Diagram

- Menggambarkan “events” dan proses yang mengelola event
- Sebuah “event” adalah kondisi boolean yang dipastikan dengan :
  - Mendaftar semua sensor yang dibaca oleh software.
  - Mendaftar semua kondisi interupsi.
  - Mendaftar semua "switches" yang dipicu oleh sebuah operator.
  - Mendaftar semua kondisi data.
  - Memanggil kembali parser kata benda/kerja yang diaplikasikan pada narasi proses, mereview semua “item kendali” sebagai input/output CSPEC.

Model Kendali

- control flow diagram berada “di atas” DFD dan menunjukkan event yang mengendalikan proses-proses yang terdapat pada DFD
- Aliran kendali—event dan item kendali– ditandai dengan panah putus-putus
- Sebuah tiang vertikal menggambarkan input menuju atau output dari sebuah control spec (CSPEC) — spesifikasi terpisah yang menggambarkan bagaimana kendali ditangani
- Sebuah panah putus-putus memasuki tiang vertikal adalah input menuju CSPEC
- Sebuah panah putus-putus meninggalkan proses menggambarkan kondisi data
- Sebuah panah putus-patas memasuki sebuah proses menggambarkan sebuah kendali input yang dibaca langsung oleh proses
- Aliran kendali tidak secara fisik mengaktifkan/menonaktifkan proses, hal ini dilakukan melalui CSPEC

Control Flow Diagram



Control Specification (CSPEC)

CSPEC dapat berupa:

- state diagram (sequential spec)
- state transition table
- decision tables
- activation tables

} combinatorial spec

Panduan Membangun CSPEC

- Lihat semua sensor yang “dibaca” oleh PL
- Lihat semua kondisi interupsi
- Lihat semua "switches" yang diaktifkan oleh operator
- Lihat semua kondisi data
- Melihat parsing kata benda/kerja yang diterapkan pada Statemen software atau lingkupnya, mereview semua item kendali Sebagai input/output CSPEC yang mungkin
- Menggambarkan perilaku sebuah sistem dengan identifikasi Keadaan; menentukan bagaimana setiap keadaan mencapai dan Menentukan transisi antar keadaan
- Fokus pada kemungkinan kehilangan/tidak tercantum... Kesalahan umum dalam menentukan kendali, misalnya "Is there any other way I can get to this state or exit from it?"

Pemodelan berbasis Class

- Tentukan analisis class dengan memeriksa pernyataan masalah(problem statement)
- Gunakan parsing gramatikal untuk memilah class potensial
- Kenali atribut tiap class
- Kenali operasi yang memanipulasi atribut-atribut tersebut

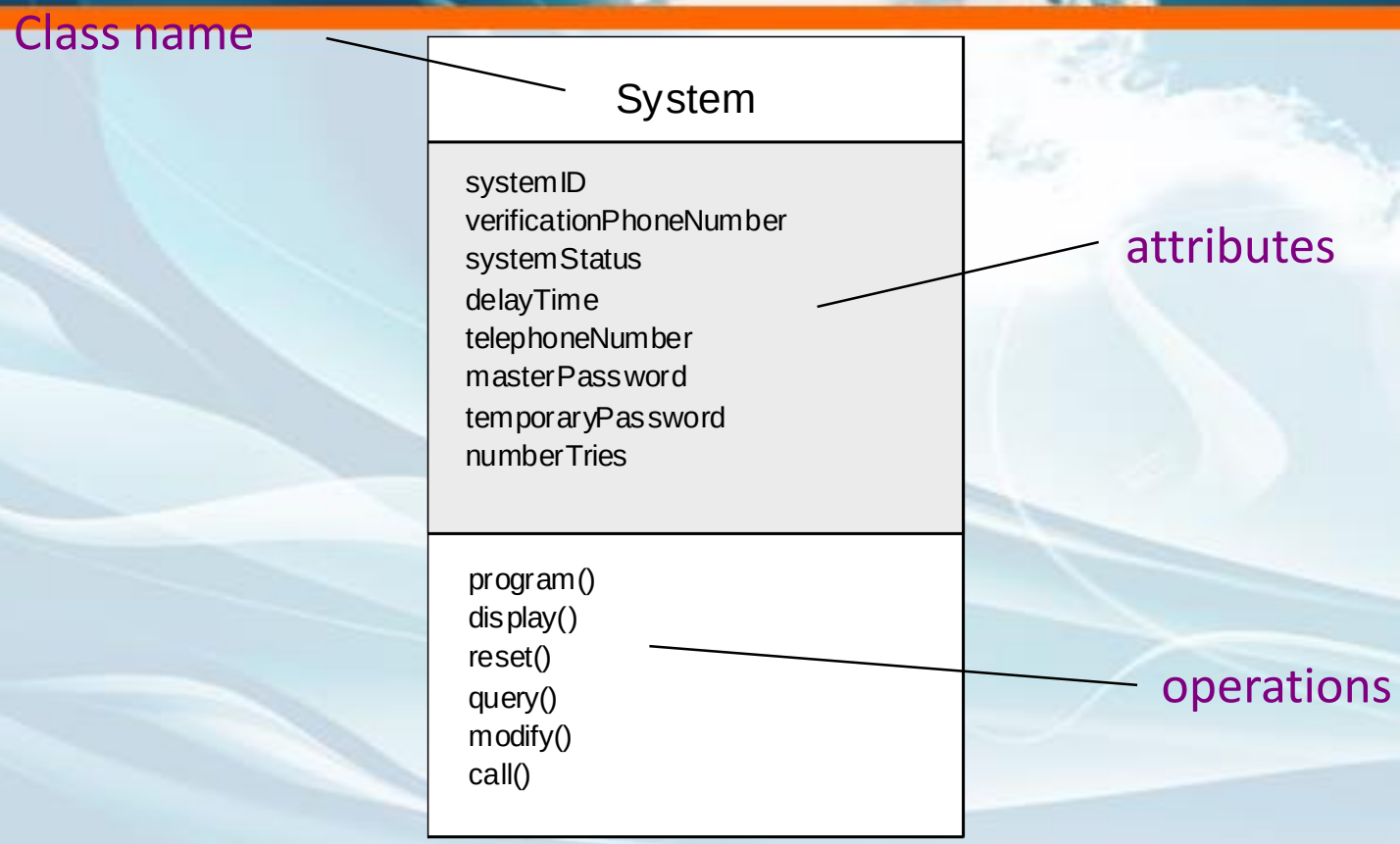
Class Analisis

- *Entitias external* (contoh : sistem lain, piranti, orang) yang menghasilkan atau menggunakan informasi yang digunakan oleh sistem berbasis komputer.
- *Benda* (contoh : laporan, display, surat, sinyal) yang merupakan bagian dari domain informasi untuk masalah.
- *Kejadian atau event* (contoh : transfer properti atau pelengkapan urutan gerakan robot) yang terjadi di dalam konteks sistem operasi.
- *Peran* (contoh : manajer, insinyur, sales) yang diperankan orang yang berinteraksi dengan sistem.
- *Unit Organisasi* (contoh : divisi, kelompok, tim) yang relevan terhadap aplikasi.
- *Tempat* (contoh : lantai pabrik, pelabuhan muatan) yang membangun konteks masalah dan fungsi keseluruhan sistem.
- *Struktur* (contoh : sensor, kendaraan 4WD, komputer) yang terdiri dari beberapa objek class atau objek-objek class yang terkait

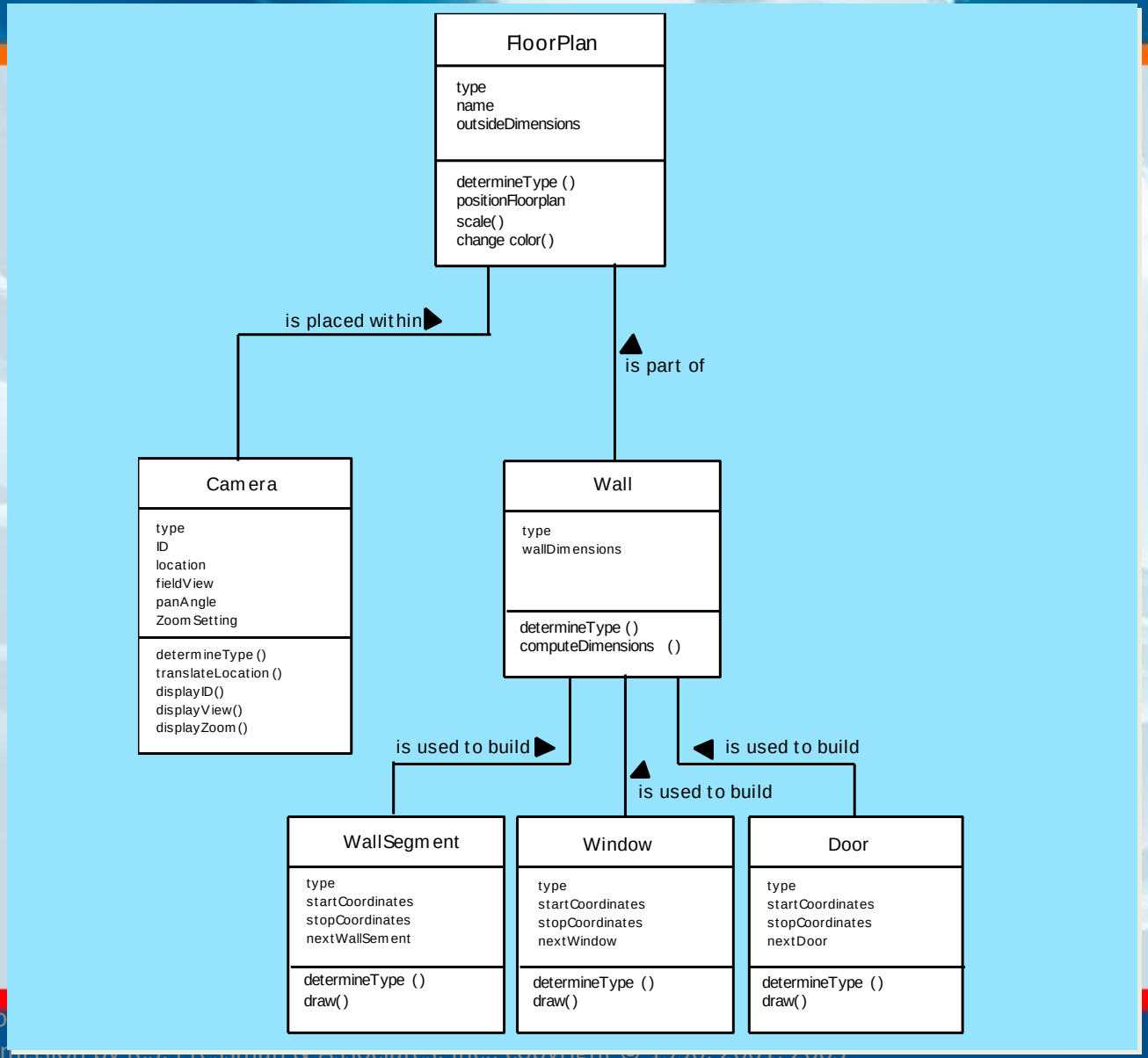
Kriteria memilih class

- Menyimpan informasi
- Layanan yang dibutuhkan
- Beberapa atribut
- Atribut umum
- Operasi umum
- Kebutuhan esensial

Class Diagram



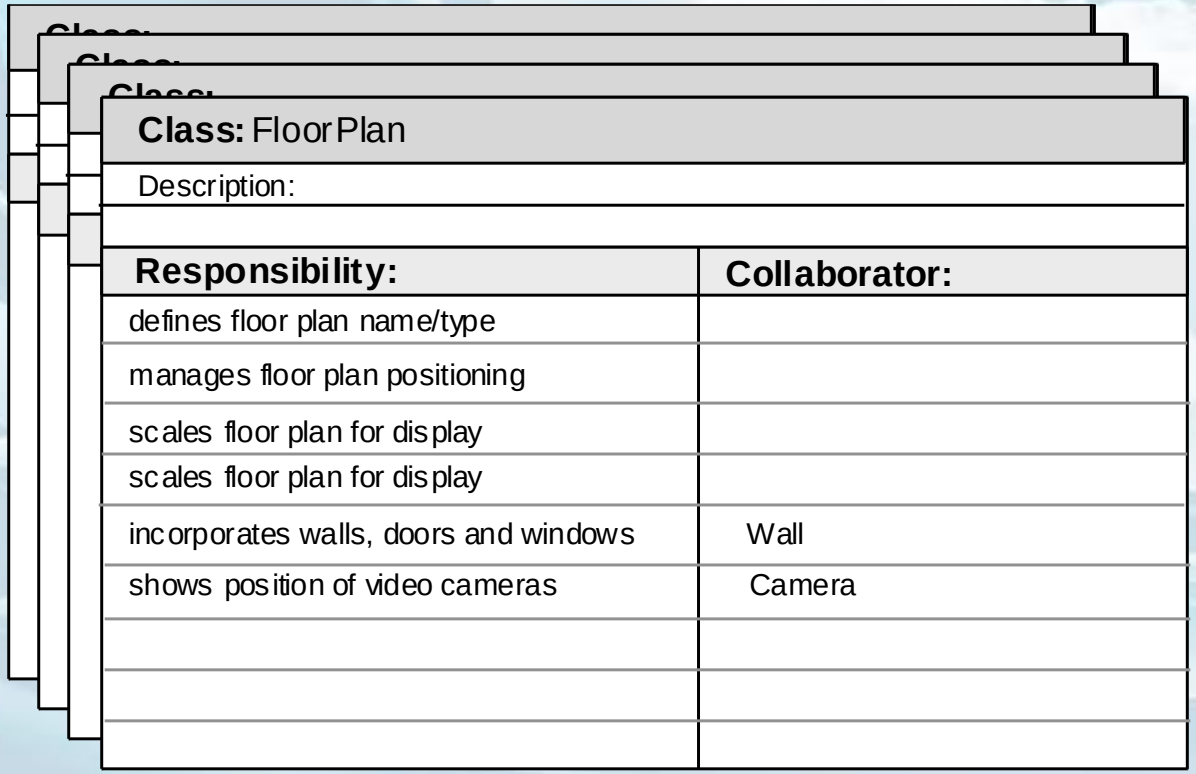
Class Diagram



Pemodelan CRC

- Class-class analisis memiliki “tanggung-jawab”
  - *Tanggungjawab* adalah atribut-atribut dan operasi-operasi yang terenkapsulasi oleh class
- Class-class analisis berkolaborasi satu dengan yang lain
  - *Collaborators* adalah class-class yang dibutuhkan untuk menyediakan sebuah class dengan informasi yang dibutuhkan untuk memenuhi tanggung jawabnya.
  - Secara umum, sebuah kolaborasi berakibat permintaan informasi atau permintaan beberapa aksi/operasi.

Pemodelan CRC



Class: FloorPlan

Description:

Responsibility:	Collaborator:
defines floor plan name/type	
manages floor plan positioning	
scales floor plan for display	
scales floor plan for display	
incorporates walls, doors and windows	Wall
shows position of video cameras	Camera

Tipe-tipe Class

- *Class entitas*, sering disebut class model atau bisnis, yang diekstrak langsung dari statemen permasalahan (contoh : Sensor).
- *Class perbatasan digunakan untuk membuat interface (contoh : layar interaktif, atau laporan cetak) dimana user melihat dan berinteraksi dengannya selama PL digunakan*.
- *Class kendali mengelola "unit kerja* [UML03] dari awal sampai akhir. Class kendali dapat didesain mengelola :
  - Pembuatan atau update objek entitas;
  - Inisiasi objek perbatasan sebagaimana mereka mendapatkan informasi dari objek entitas;
  - Komunikasi kompleks antara sekumpulan objek;
  - Validasi data yang dikomunikasikan antara user dan aplikasi.

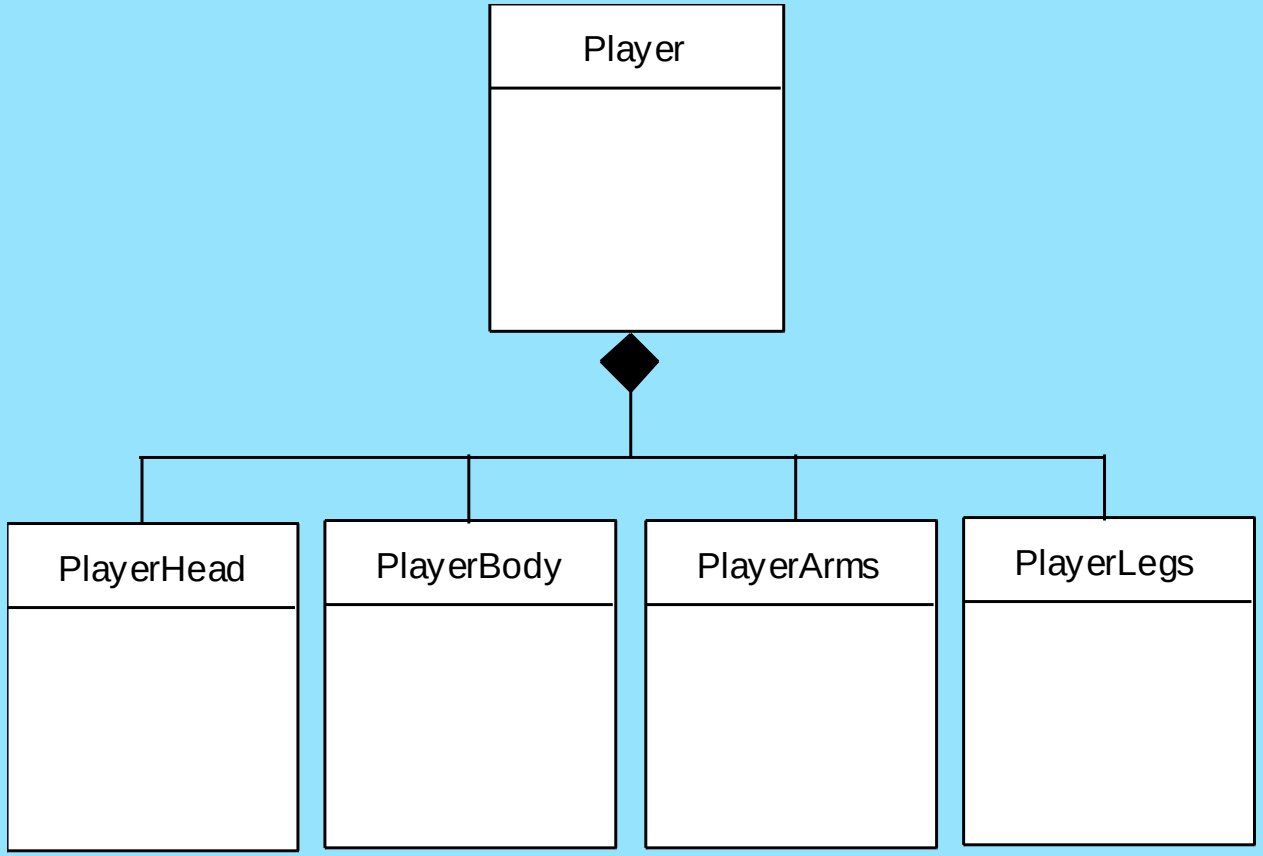
Responsibilities

- System intelligence should be distributed across classes to best address the needs of the problem
- Each responsibility should be stated as generally as possible
- Information and the behavior related to it should reside within the same class
- Information about one thing should be localized with a single class, not distributed across multiple classes.
- Responsibilities should be shared among related classes, when appropriate.

Kolaborasi

- Class memenuhi tanggung jawabnya dengan satu diantara dua cara :
  - Sebuah class dapat menggunakan operasinya sendiri untuk memanipulasi atributnya masing-masing atau
  - Sebuah class dapat berkolaborasi dengan class lainnya.
- Kolaborasi membuat relasi antara class
- Kolaborasi dapat diidentifikasi dengan menentukan apakah sebuah class dapat memenuhi tanggung jawabnya masing-masing
- Tiga relasi umum yang berbeda antar class [WIR90]:
  - *is-part-of* relationship
  - *has-knowledge-of* relationship
  - *depends-upon* relationship

Composite Aggregate Class



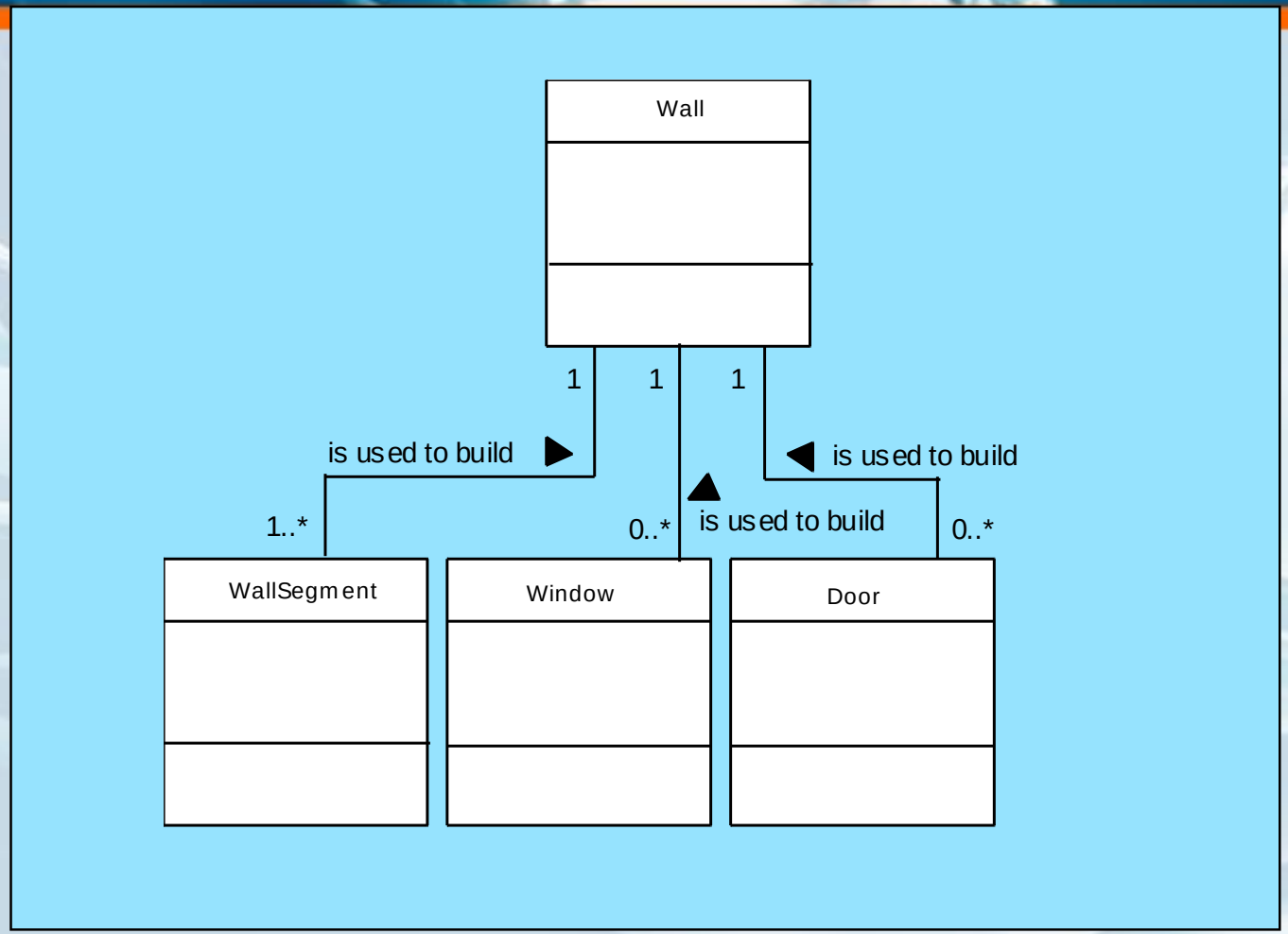
Review model CRC

- Semua peserta dalam review (model CRC) diberikan sebuah subset dari kartu index model CRC.
  - Kartu yang berkolaborasi harus terpisah (tidak boleh ada reviewer yang memiliki dua kartu yang berkolaborasi).
- Semua skenario use-case (dan diagram use case terkait) harus diorganisasi dalam kategori-kategori.
- Pemimpin review membaca use-case secara hati-hati.
  - Ketika pemimpin review sampai pada objek, dia akan memberi tanda kepada person yang memegang kartu index class yang terkait.
- Ketika tanda dikirimkan, pemilik kartu class diminta untuk menggambarkan tanggung jawab yang tertulis di kartu tersebut.
  - Kelompok menentukan satu (atau lebih) tanggung jawab yang memenuhi kebutuhan use-case.
- Jika tanggung jawab dan kolaborasi yang tertera pada kartu index tidak dapat mengakomodasi use-case, modifikasi dilakukan pada kartu tersebut.
  - Hal ini termasuk definisi class baru (dan kartu index CRC) atau spesifikasi baru atau revisi mengenai tanggung jawab, atau kolaborasi kartu yang sudah ada.

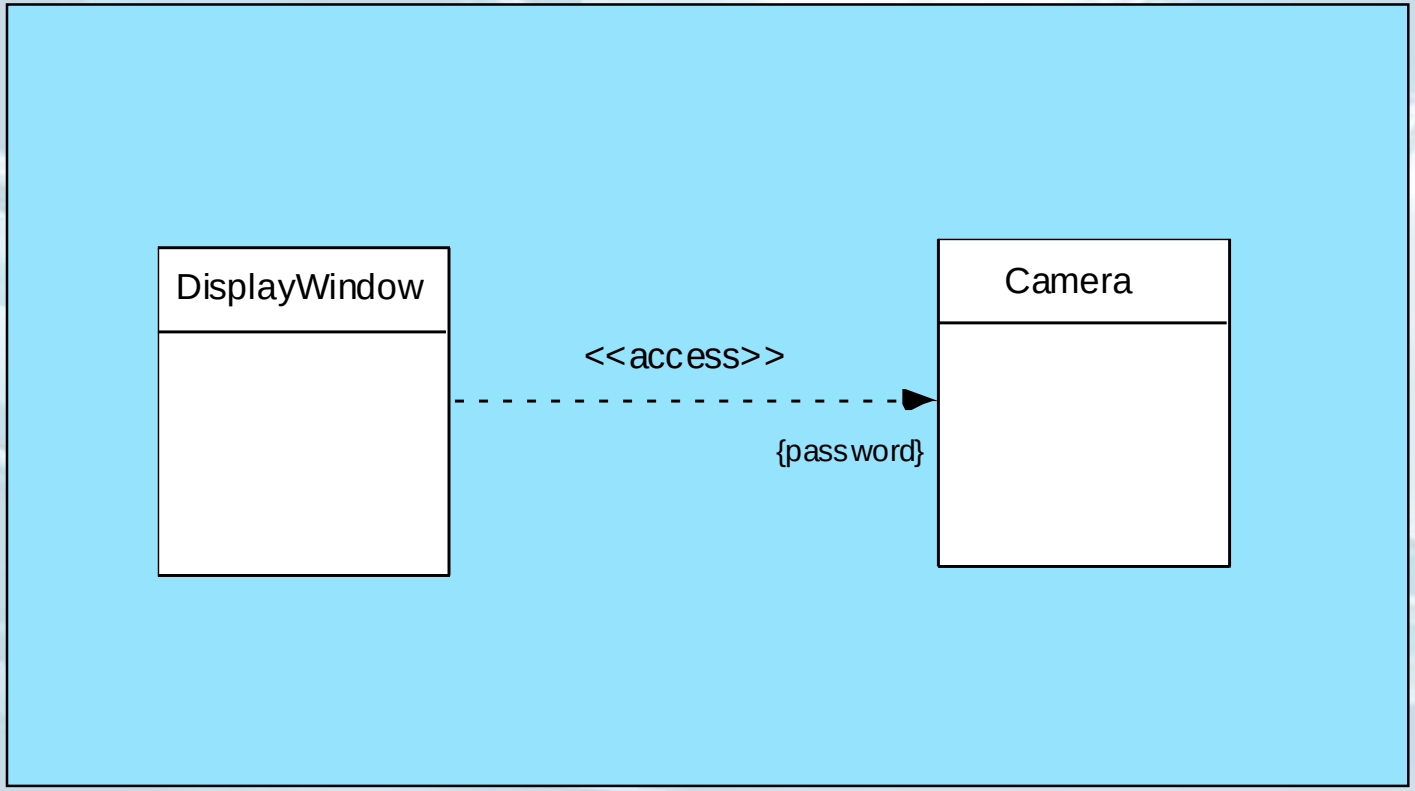
Asosiasi dan Dependensi

- Dua class analisis sering berhubungan satu dengan yang lain dalam beberapa pola
  - Dalam UML relasi ini sering disebut *asosiasi*
  - Asosiasi dapat didapatkan dengan mengenali *multiplicity* (istilah *cardinality* digunaikan dalam pemodelan data
- Dalam banyak instans, relasi client-server ada diantara dua class analisis.
  - Dalam kasus ini, class client tergantung pada class server dalam suatu cara dan relasi dependensi terjadi

Multiplicity



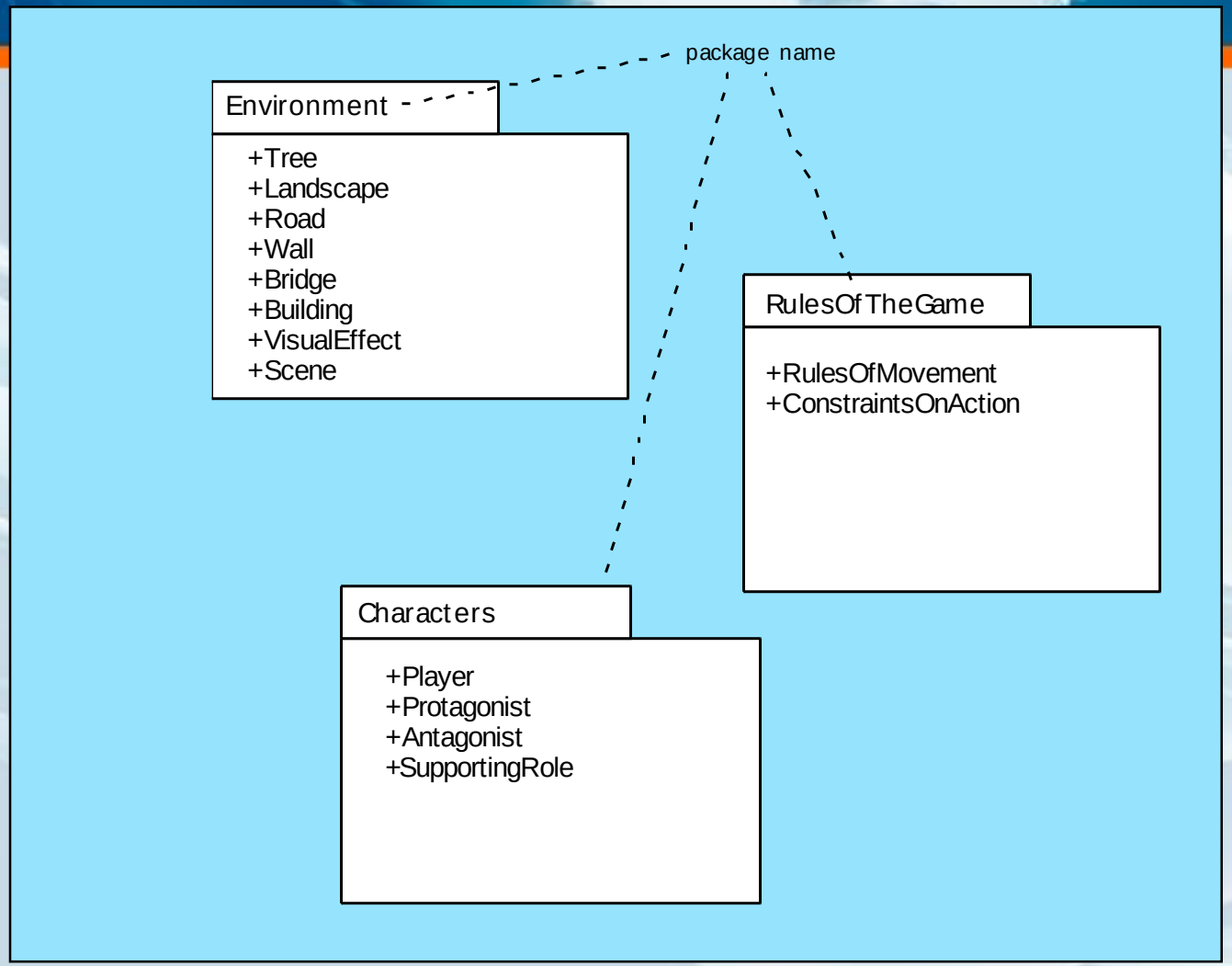
Dependencies



Analisis Paket

- Beberapa model analisis (use-case, class analisis) dikategorisasi dalam sebuah pola yang mempaketkan mereka dalam kelompok
- Tanda plus di dalam nama class analisis dalam setiap paket menandakan bahwa class-class tersebut mempunyai visibilitas publik dan karena itu dapat diakses dari paket lain.
- Simbol lain dapat mendahului elemen di dalam paket. Tanda minus menandakan bahwa elemen disembunyikan dari semua paket, dan tanda # menandakan bahwa elemen hanya dapat diakses oleh paket yang berada di dalam paket tersebut.

Analysis Packages



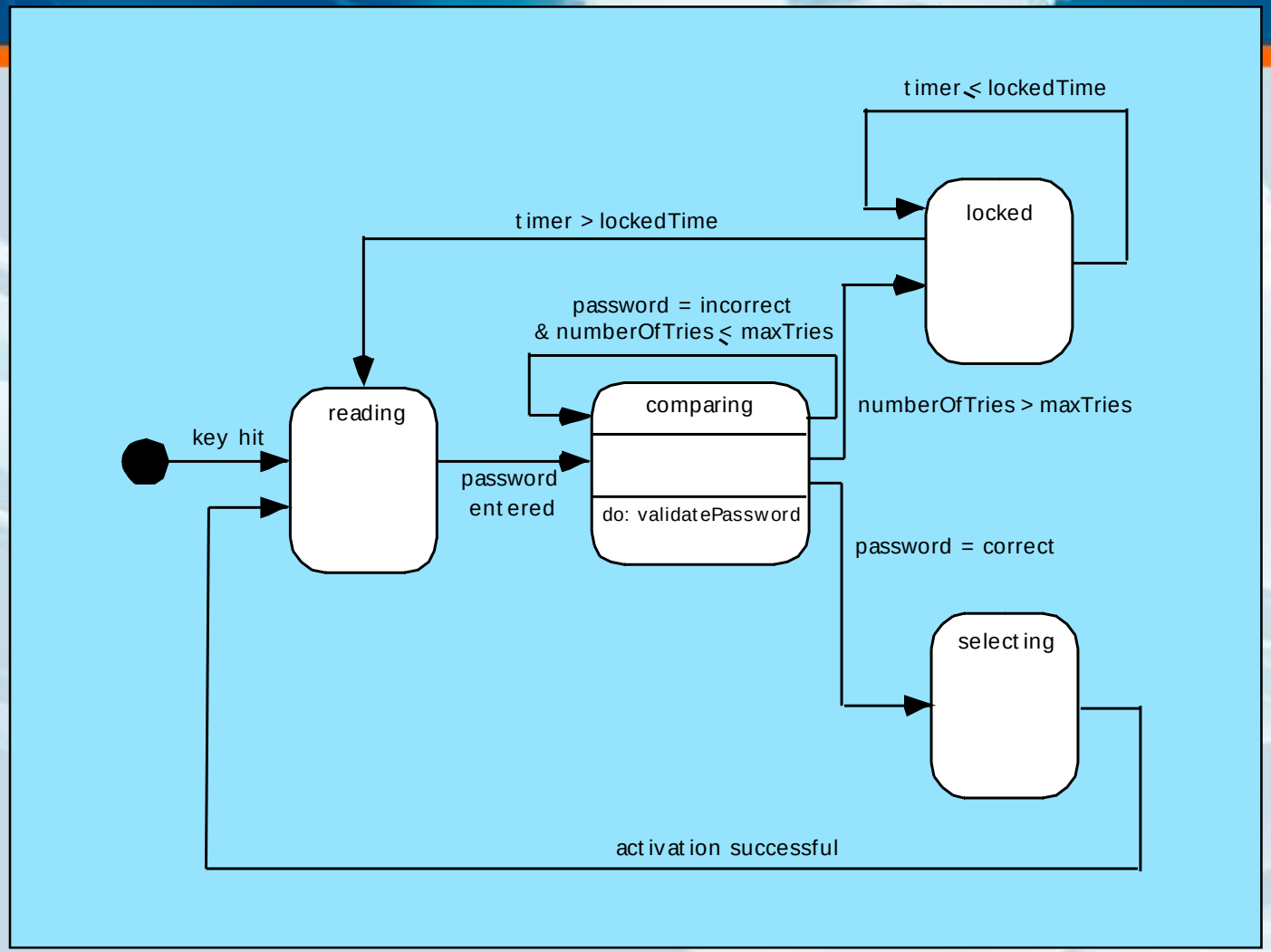
Pemodelan Perilaku

- Model perilaku menggambarkan bagaimana PL merespon event atau stimulan eksternal. Untuk model tersebut, analis harus melakukan langkah-langkah berikut :
  - Evaluasi semua use-case untuk mendapatkan pemahaman menyeluruh tentang urutan interaksi di dalam sistem.
  - Mengenali event yang mengendalikan urutan interaksi dan memahamibagaimana event mempunyai relasi terhadap objek spesifik.
  - Membuat urutan untuk setiap use-case.
  - Membangun state diagram untuk sistem.
  - Review model behavioral untuk memverifikasi akurasi dan konsistensi

Representasi Keadaan

- Dalam konteks pemodelan perilaku, dua karakter keadaan harus diperhatikan :
  - Keadaan setiap class ketika sistem menjalankan fungsinya, dan
  - Keadaan sistem ketika diobservasi dari luar sebagaimana sistem menjalankan fungsinya.
- Keadaan class mengambil baik karakter aktif maupun pasif [CHA93].
  - Sebuah *keadaan pasif* adalah status saat ini dari semua atribut objek.
  - *Keadaan aktif* dari sebuah objek menggambarkan status saat ini pada objek tersebut ketika menjalankan transformasi atau proses.

State Diagram for the ControlPanel Class



Keadaan-Keadaan Sistem

- state—sekumpulan keadaan terobservasi yang menggambarkan perilaku sistem pada satu waktu
- state transition—perubahan dari satu keadaan ke keadaan yang lain
- event—sebuah kejadian yang menyebabkan sistem melakukan perilaku yang sudah diprediksi sebelumnya
- action—proses yang terjadi sebagai konsekuensi membuat transisi

Pemodelan Perilaku

- Membuat daftar keadaan sistem yang berbeda (Bagaimana perilaku sistem ?)
- Menggambarkan bagaimana sistem membuat transisi dari satu keadaan ke keadaan yang lain.indicate how the system makes a transition from one state to another (Bagaimana sistem mengubah keadaan?)
  - mengenali event
  - Mengawali action
- Menggambar sebuah state diagram atau sequence diagram

Sequence Diagram

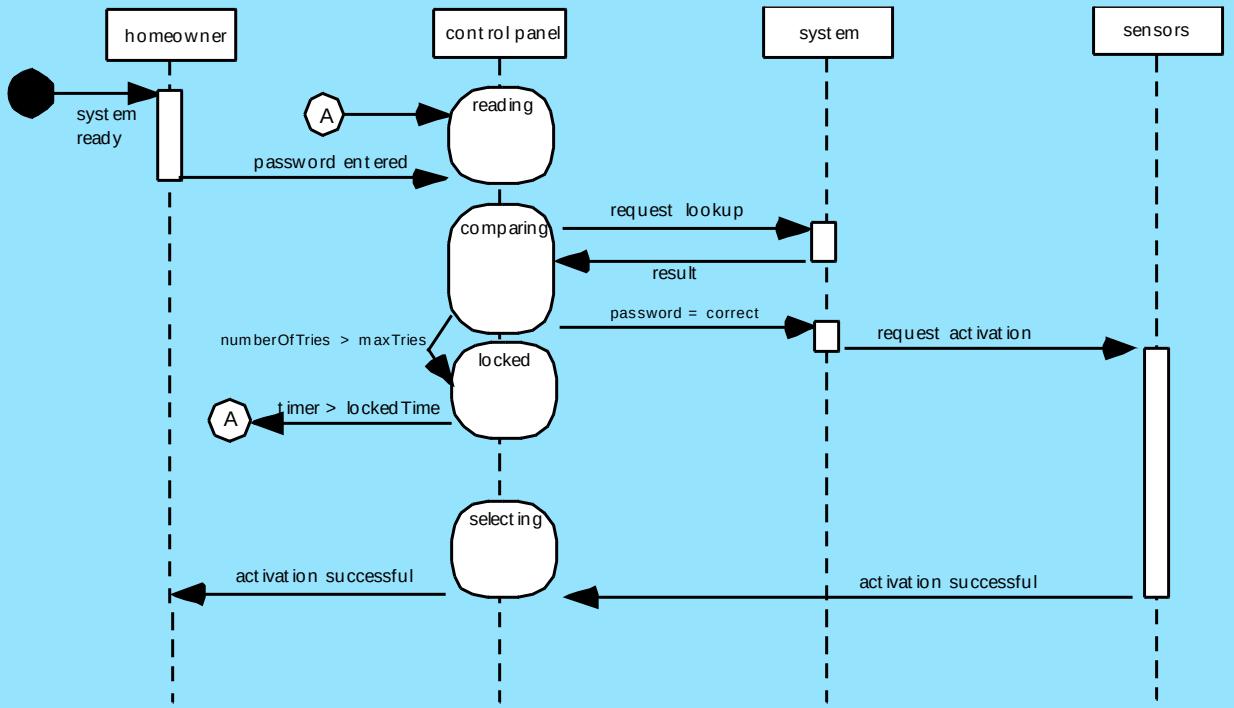


Figure 8.27 Sequence diagram (partial) for *SafeHome* security function

Menulis Spesifikasi PL