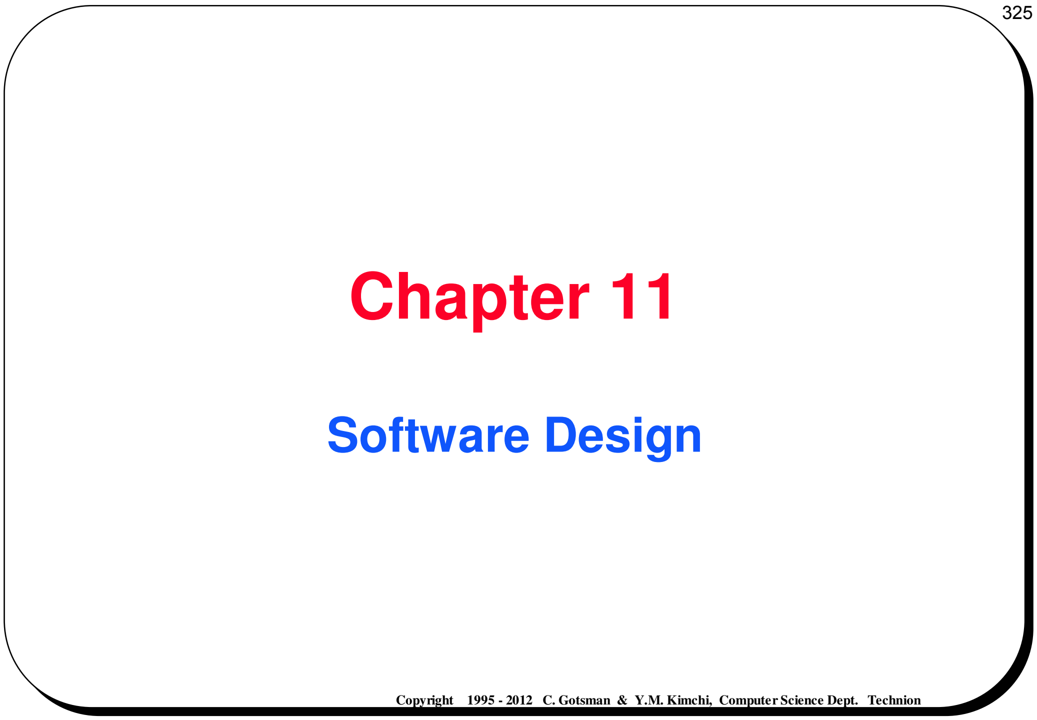
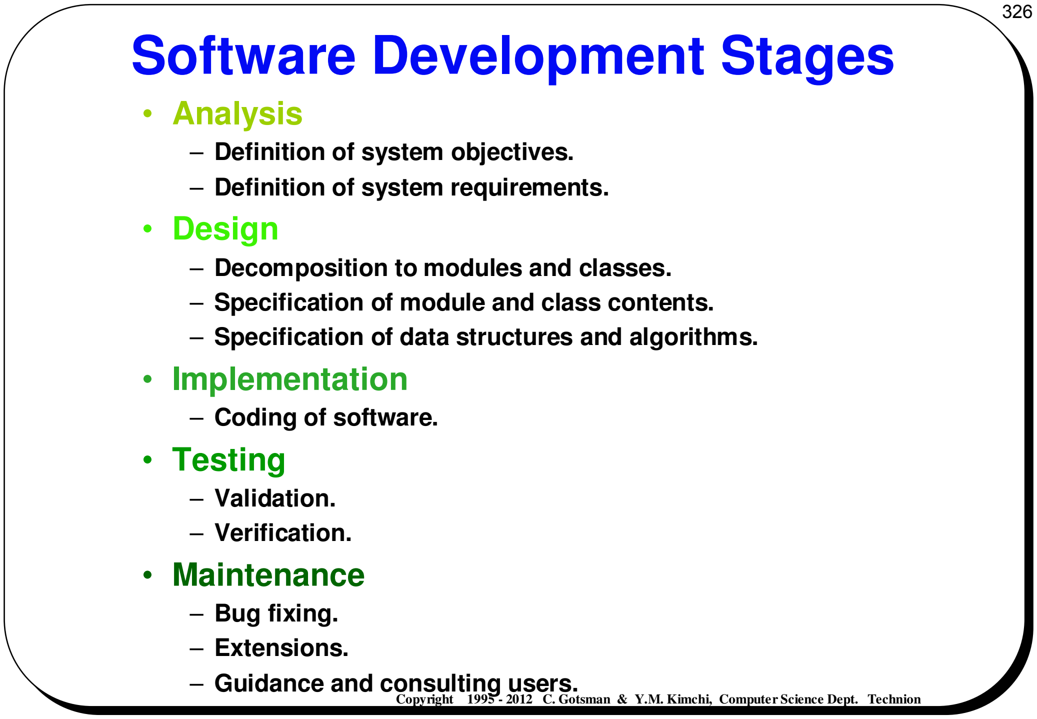


325

Chapter 11

Software Design



326

Software Development Stages

- Analysis
  - Definition of system objectives.
  - Definition of system requirements.
- Design
  - Decomposition to modules and classes.
  - Specification of module and class contents.
  - Specification of data structures and algorithms.
- Implementation
  - Coding of software.
- Testing
  - Validation.
  - Verification.
- Maintenance
  - Bug fixing.
  - Extensions.
  - Guidance and consulting users.

The Object Oriented Paradigm

- About 70% of software projects fail
- In many cases the tool works but is useless
  - Does not meet customers' needs
  - Too late for the market
- The OO paradigm is addressed at overcoming software complexity
  - Make the tool easy to comprehend (user & developer)
  - Adhere to the *open-close* principle at each and every level (from class-level and up)
  - The above two result in significantly ease modifications and extensions
- Object Orientation is about *process*
  - It is not about *design* or about *programming*
  - It is not about *classes* or about *packages*
  - It is not about *UML* or about *C++*

Requirement Analysis

- Detailed interviews and dialogues with customers
  - Customers may have conflicting and/or contradictory expectations
- Textual descriptions of all various usage modes are created
  - These describe in details communication among participants
  - These are called *scripts*
  - Participants are called *actors*
- These descriptions are called

  Use Cases

329

Unified Modeling Language

- Based on previously used practices
- A Collection of various modeling techniques
- Allow modeling of different aspects
- They were modified to have a common style

330

Use Cases

- A use case is the basis for
  - Use case diagrams
  - Collaboration diagrams
  - Sequence diagrams
- Eventually, use cases are the bases for the software
  - Nouns become classes
  - Verbs become messages
  - Relationships become relations
- Relationships between use cases
  - << include >> *(common to several)*
  - << extend >> *(a possible extension)*
  - generalization *(of a more specific)*

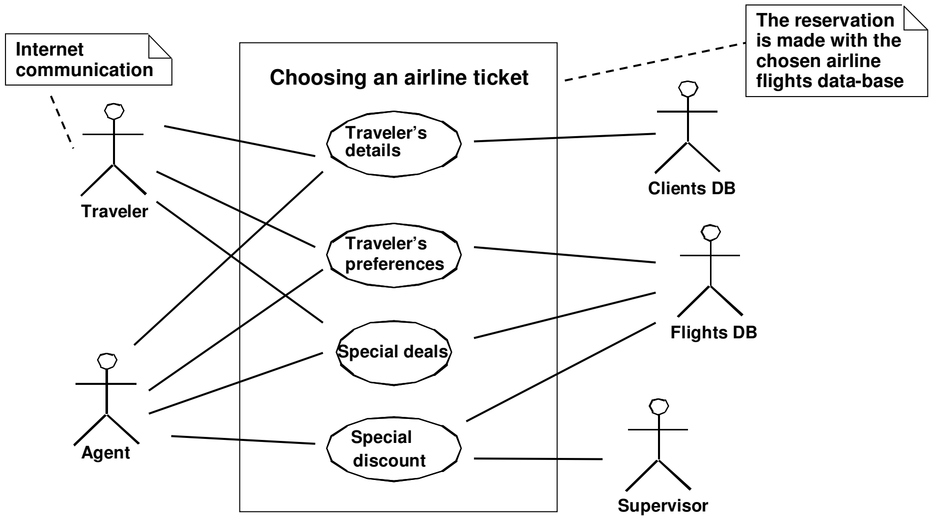
331

A Use Case Description:
a traveler orders an airline ticket

- **Actors:** traveler, travel agent, supervisor, flights DB (customer DB)
- **Preconditions:** cummunication channels established
- **Primary course**
  - Traveler presents personal details *(optional with an agent)*
  - Traveler presents goals: date, destination, price, airline
  - Traveler prioritize goals and presents constraints
  - Agent checks for special deals: student, frequent-flier . . .
  - Agent presents 3 best matches, then 10 best matches
  - Traveler negotiates: price-difference, departure time, . . .
  - . . . . . .
- **Secondary courses**
  - Agent asks supervisor for a special discount
  - Usecase needs no agent if traveler connects to DB via Internet
    - » No special discount is available in this course
    - » Traveler sees less DB-information on screen

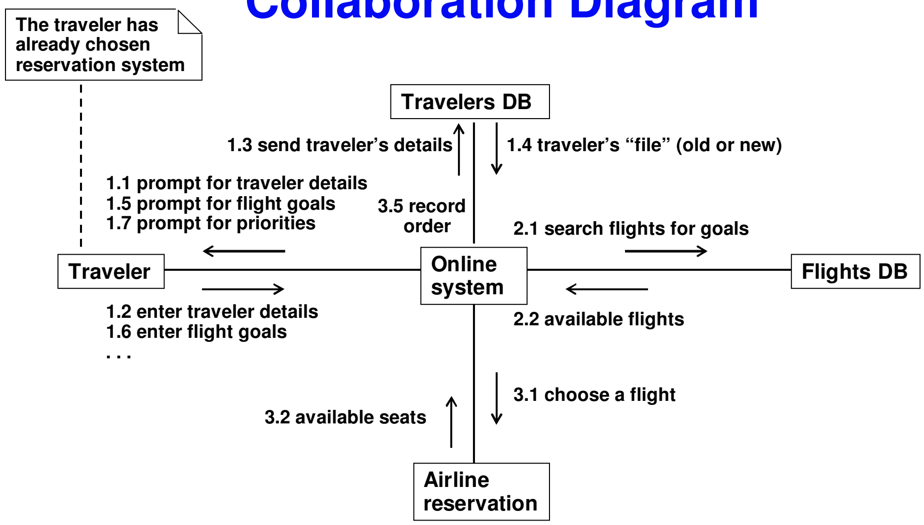
332

Use Case Diagram



333

Collaboration Diagram



The traveler has already chosen reservation system

Travelers DB

Traveler

Online system

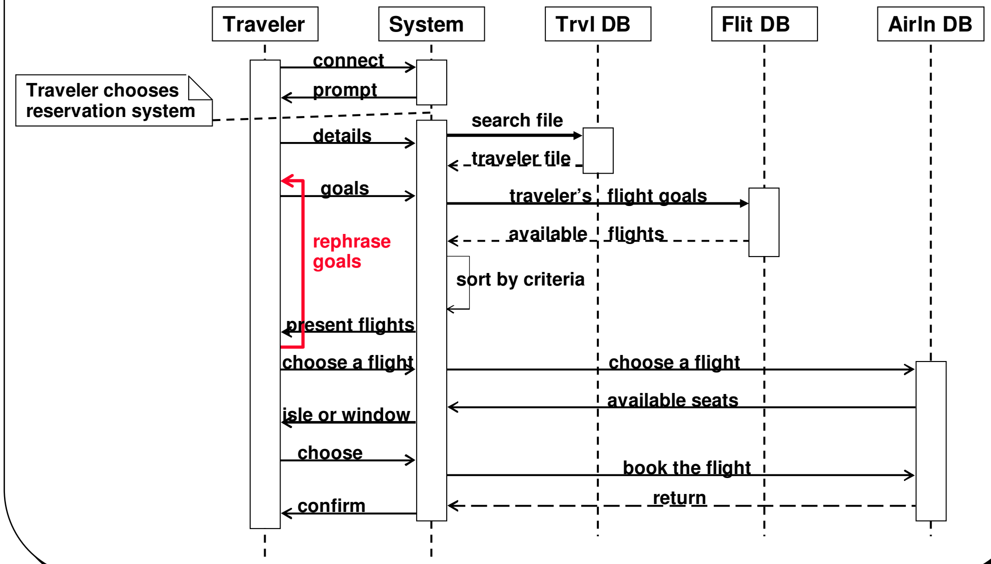
Flights DB

Airline reservation

1.1 prompt for traveler details
1.2 enter traveler details
1.3 send traveler’s details
1.4 traveler’s “file” (old or new)
1.5 prompt for flight goals
1.6 enter flight goals
1.7 prompt for priorities
. . .
2.1 search flights for goals
2.2 available flights
3.1 choose a flight
3.2 available seats
3.5 record order

334

Sequence Diagram



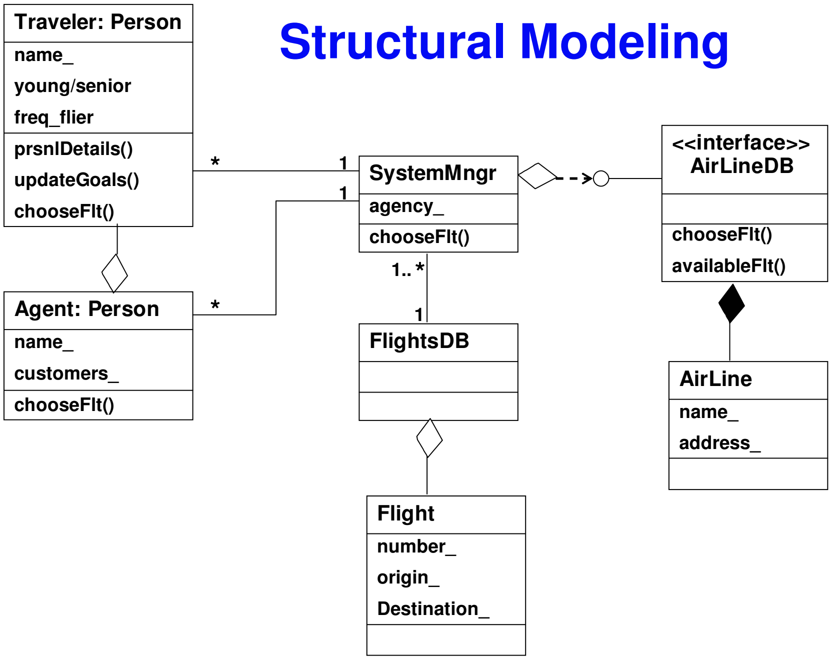
Traveler | System | Trvl DB | Flit DB | Airln DB

Traveler chooses reservation system

connect
prompt
details
search file
traveler file
goals
traveler’s flight goals
available flights
rephrase goals
sort by criteria
present flights
choose a flight
choose a flight
available seats
isle or window
choose
book the flight
return
confirm

335

Structural Modeling



Traveler: Person
name_
young/senior
freq_flier
prsnlDetails()
updateGoals()
chooseFlt()

Agent: Person
name_
customers_
chooseFlt()

SystemMngr
agency_
chooseFlt()

FlightsDB

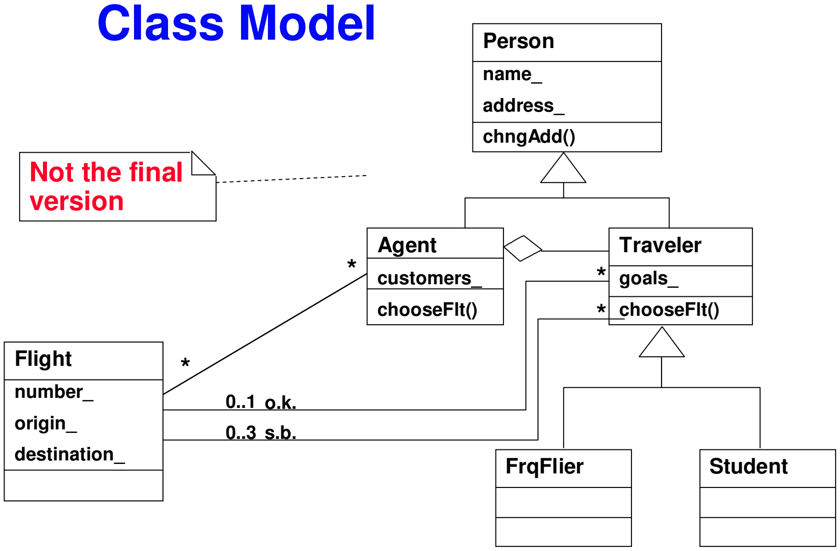
Flight
number_
origin_
Destination_

<<interface>>
AirLineDB
chooseFlt()
availableFlt()

AirLine
name_
address_

336

Class Model



Not the final version

Person
name_
address_
chngAdd()

Agent
customers_
chooseFlt()

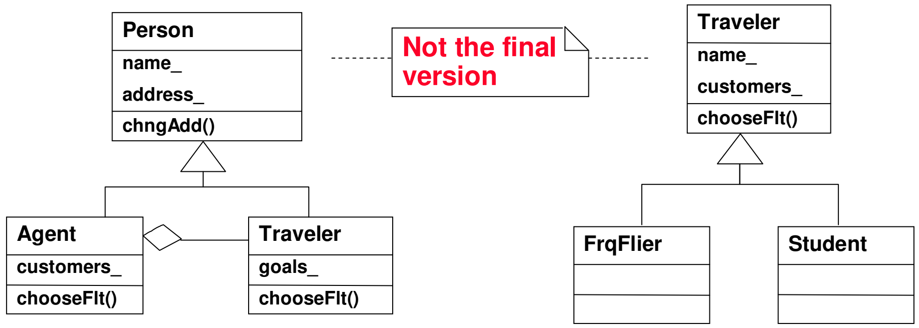
Traveler
goals_
chooseFlt()

Flight
number_
origin_
destination_

FrqFlier

Student

Class Model (cont.)

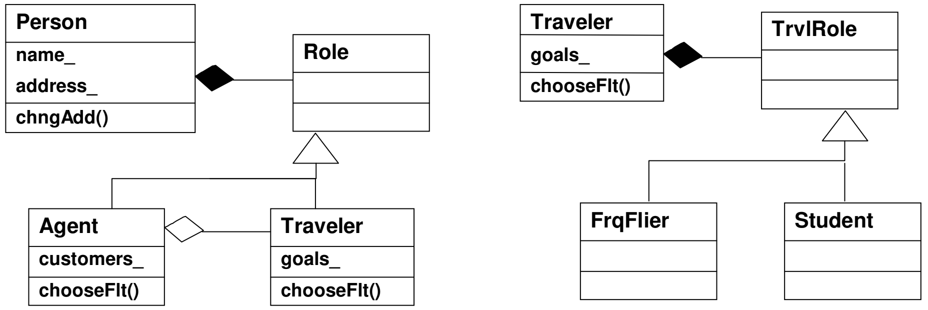


Not the final version

How can an agent become a traveler? become both?

How can a student become a FrqFlier? become both?

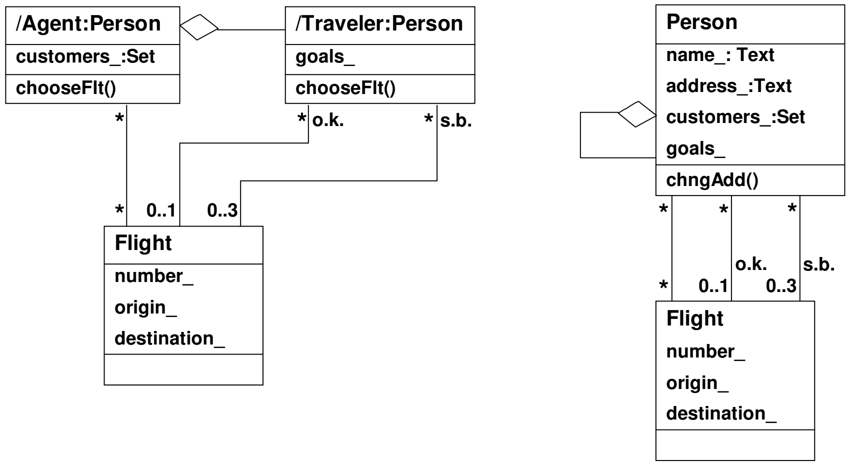
Class Model (cont.)



Here we touch the domain of Design Patterns

A *pattern* is a solution to an abstract problem

Role Model vs. Class Model



/Agent:Person
customers_:Set
chooseFlt()

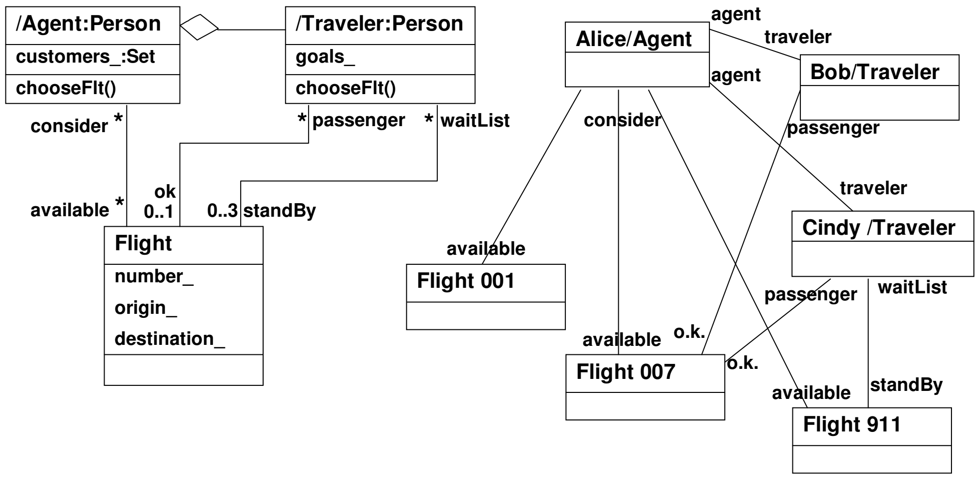
/Traveler:Person
goals_
chooseFlt()

Flight
number_
origin_
destination_

Person
name_: Text
address_:Text
customers_:Set
goals_
chngAdd()

Flight
number_
origin_
destination_

Collaboration Diagram
Specification Level vs. Instance Level



/Agent:Person
customers_:Set
chooseFlt()

/Traveler:Person
goals_
chooseFlt()

Flight
number_
origin_
destination_

Alice/Agent

Bob/Traveler

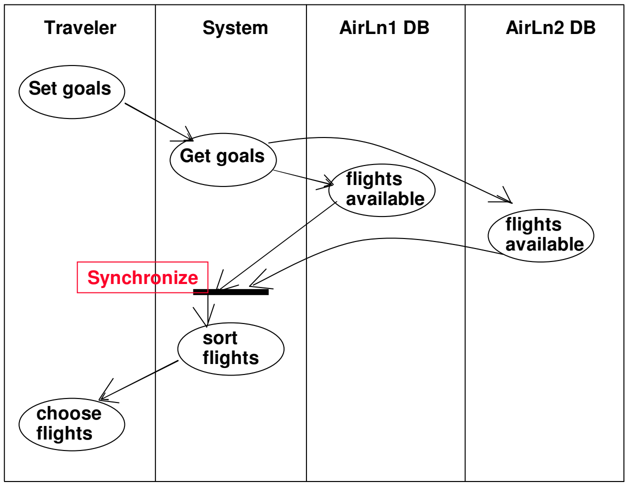
Cindy /Traveler

Flight 001

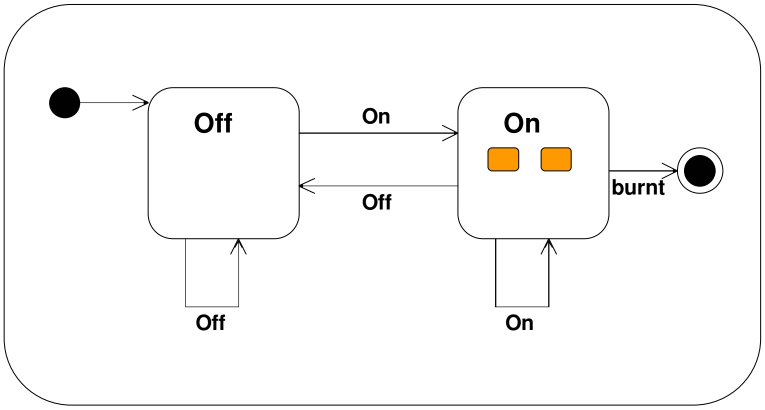
Flight 007

Flight 911

Activity Diagram

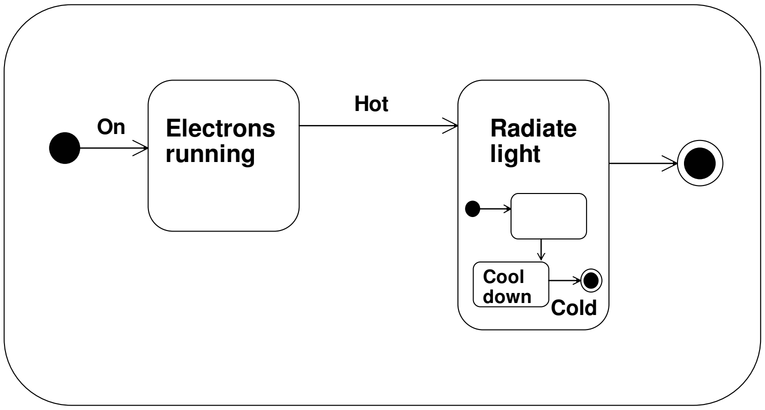


Statecharts



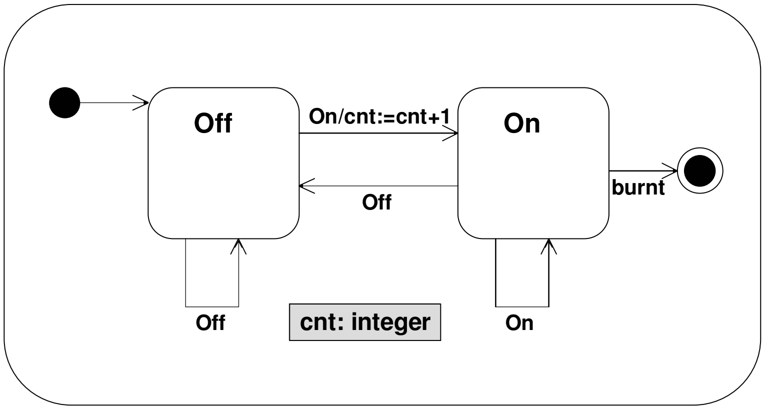
A light bulb

Statecharts (cont.)



On (A light bulb)

Statecharts



A light bulb with a variable

More Construct

- A few examples
- An old example

Associations

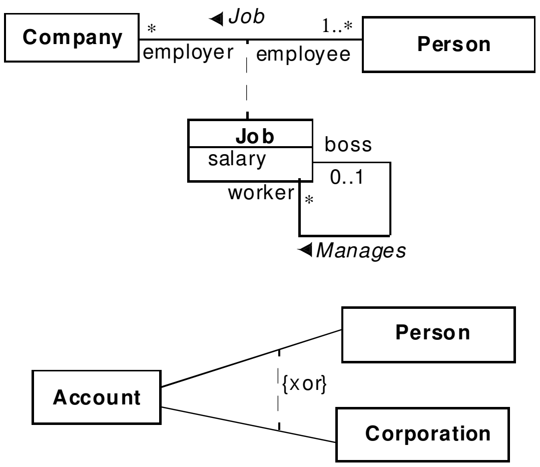


Fig. 3-31, *UML Notation Guide*

Association Ends

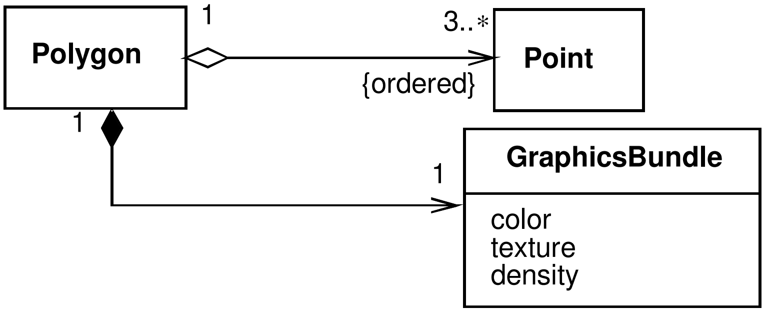


Fig. 3-32, *UML Notation Guide*

Composition 1, 2

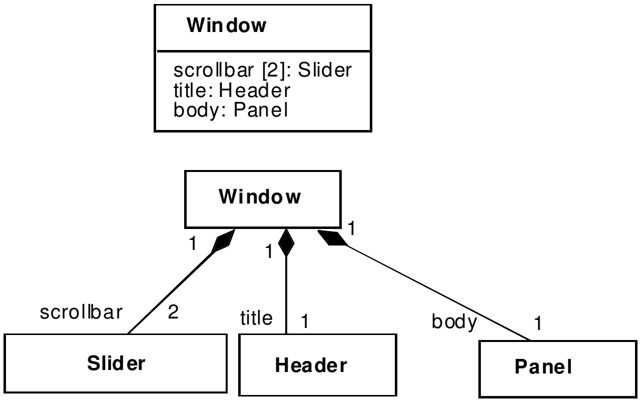
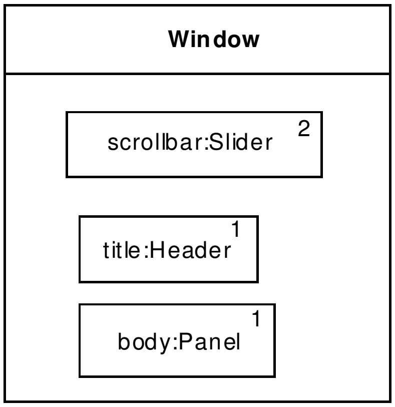


Fig. 3-36, *UML Notation Guide*

349

Composition 3



Window

scrollbar:Slider 2

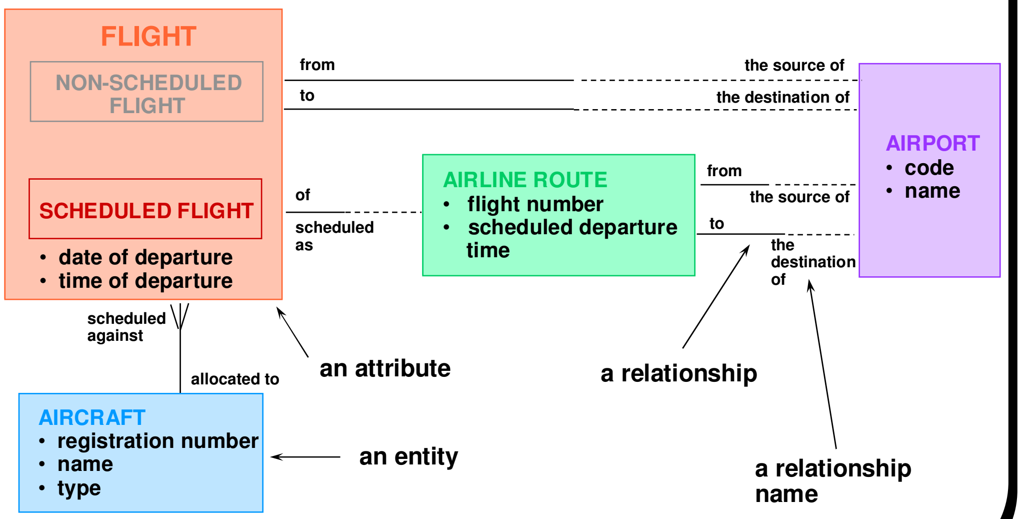
title:Header 1

body:Panel 1

Fig. 3-36, *UML Notation Guide*

350

Entity-Relationship Diagram (pre-UML)



FLIGHT

NON-SCHEDULED FLIGHT

SCHEDULED FLIGHT

- date of departure
- time of departure

from

to

the source of

the destination of

AIRPORT

- code
- name

of

scheduled as

AIRLINE ROUTE

- flight number
- scheduled departure time

from

the source of

to

the destination of

scheduled against

allocated to

AIRCRAFT

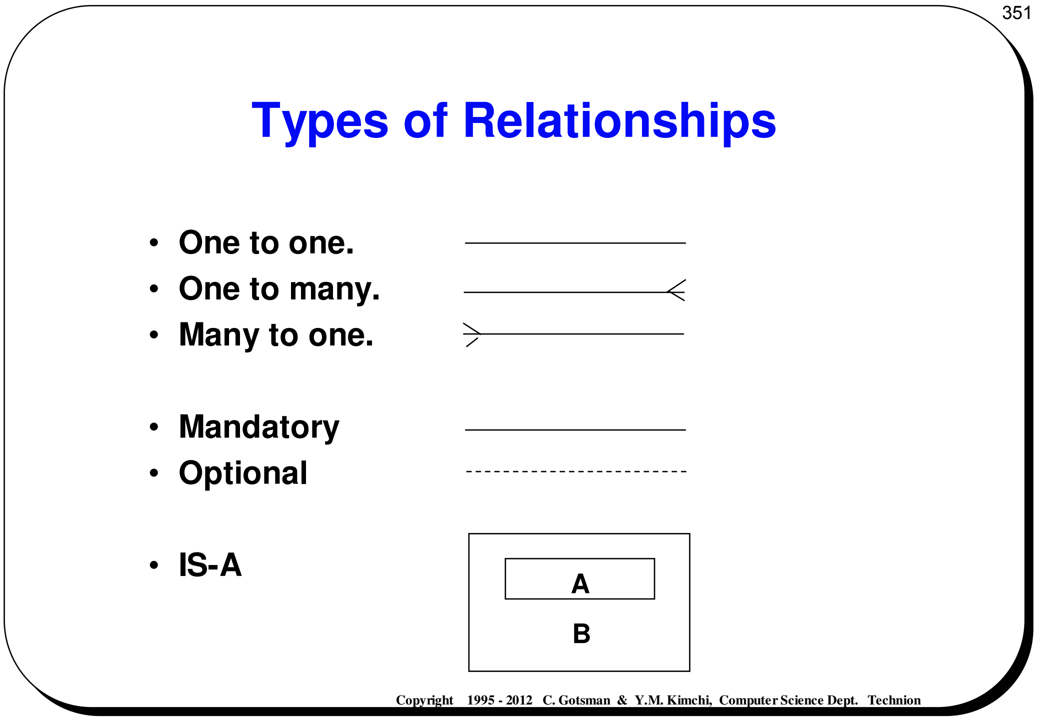
- registration number
- name
- type

an attribute

a relationship

an entity

a relationship name



351

Types of Relationships

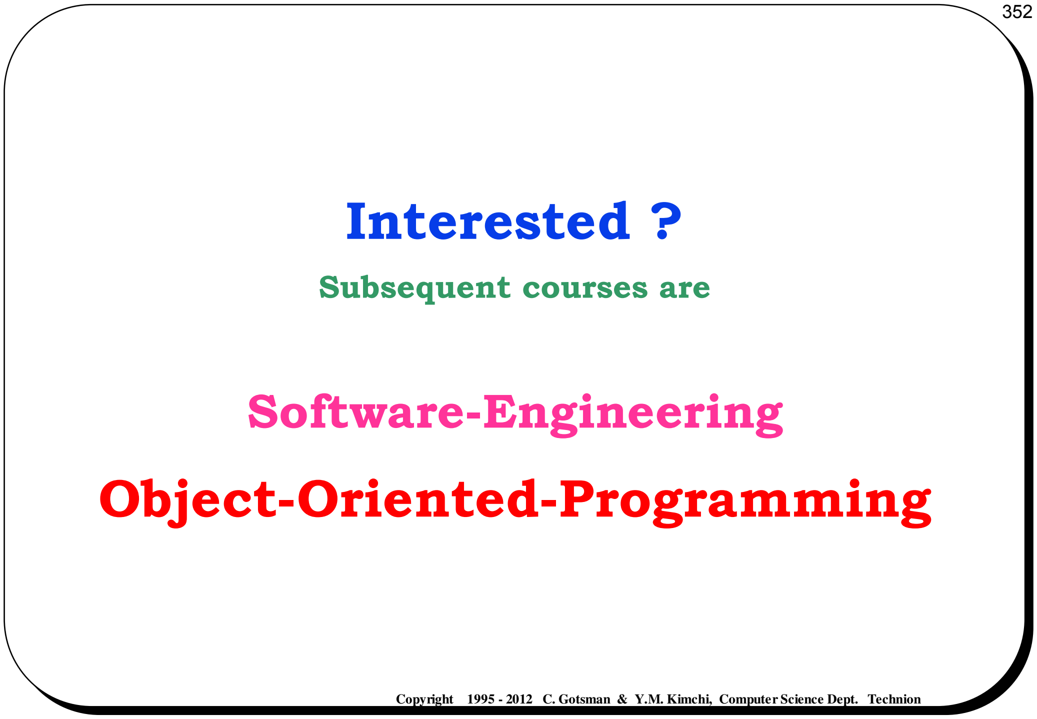
- One to one.
- One to many.
- Many to one.

- Mandatory
- Optional

- IS-A

A

B



352

Interested ?

Subsequent courses are

Software-Engineering

Object-Oriented-Programming