University of
Southampton

COMP1202 – Object Oriented Design

Designing Applications

Son Hoang
(adapted from Prof David Millard’s slides)

COMP1202 (AY2022-23)

On Building Better Classes (Recap)

- Object-oriented Techniques
  - Encapsulation: A class should be responsible for managing itself
  - Inheritance: Super- and sub-classes
  - Polymorphism: Substitution, overriding, and dynamic binding
- Error Handling:
  - Print and default
  - Error codes
  - Exceptions
- Debugging: Syntax vs logical errors
- Testing strategies: Equivalence classes and boundary value

On Building Better Classes (Recap)

- Duplication
- Coupling
- Cohesion
- Responsibility-Driven Design
- Refactoring

Coming Up

- Analysis and Design
  - Noun Verb Analysis
- Software Engineering
- Design Patterns

Part 1

Analysis and Design

The Noun/Verb Method

- Given a written problem – identifying the nouns and verbs can help to reveal the potential classes, data and methods

- The nouns in a description refer to ‘things’.
  - A source of classes and objects.
- The verbs refer to actions.
  - A source of interactions between objects.
  - Actions are behaviour, and hence methods.

Noun Phrase Parsing

- In order to find the key objects and actions
  - Search through the problem definition and
  - extract all the noun phrases
- Noun phrases are phrases which describe, individuate or pick-out things in the world
  - for example, "customer" individuates an entity which will be represented in the system
- Don't worry about whether or not the noun phrases should be part of the final solution, just meticulously list the noun phrases.

A Problem Description

The cinema booking system should store seat bookings for multiple theatres.

Each theatre has seats arranged in rows.

Customers can reserve seats and are given a row number and seat number.

They may request bookings of several adjoining seats.

Each booking is for a particular show (i.e., the screening of a given movie at a certain time).

Shows are at an assigned date and time and scheduled in a theatre where they are screened.

The system stores the customers’ telephone numbers.

Nouns?

The cinema booking system should store seat bookings for multiple theatres.

Each theatre has seats arranged in rows.

Customers can reserve seats and are given a row number and seat number.

They may request bookings of several adjoining seats.

Each booking is for a particular show (i.e., the screening of a given movie at a certain time).

Shows are at an assigned date and time and scheduled in a theatre where they are screened.

The system stores the customers’ telephone numbers.

Verb Phrase Parsing

- In order to find the common processes, look for verb phrases:
  - those which describe "doing things",
  - for example “store” is a process which summarises part of the process

- Don't worry about whether or not the verb phrases describe the final processes of the system, or whether or not one subsumes the description of the other, **just list them**.

Verbs?

The cinema booking system should store seat bookings for multiple theatres.

Each theatre has seats arranged in rows.

Customers can reserve seats and are given a row number and seat number.

They may request bookings of several adjoining seats.

Each booking is for a particular show (i.e., the screening of a given movie at a certain time).

Shows are at an assigned date and time and scheduled in a theatre where they are screened.

The system stores the customers' telephone numbers.

Verbs?

The cinema booking system should store seat bookings for multiple theatres.

Each theatre has seats arranged in rows.

Customers can reserve seats and are given a row number and seat number.

They may request bookings of several adjoining seats.

Each booking is for a particular show (i.e., the screening of a given movie at a certain time).

Shows are at an assigned date and time and scheduled in a theatre where they are screened.

The system stores the customers' telephone numbers.

Tidy up the Lists

- Most often, the requirements will be from a *domain of discourse* or "mini-world" -- a given requirements specification will be in the language of a particular work practice, such as hospitality. Given this, you can:
  - **remove *synonyms*** (noun phrases which mean the same thing in the domain of discourse).
  - **Ignore pronouns** and articles such as “the”, because they refer to an object/noun phrase in the context of the rest of the sentences.

Sketch Processes

- Look for Noun Verb pairs
  - Reserve Seats
  - Request Booking
- The processes may be described at different levels of detail
  - E.g. Store Booking is part of Reserve Seats
- Figure out which noun-verb pairs are parts of another
- But Beware!
  - Sometimes there will be a high-level phrase (Reserve Seats)
  - But sometimes there won’t be
    - Invent one by grouping together the lower-level phrases

Nouns-Verb Phrases?

The cinema booking system should store seat bookings for multiple theatres.

Each theatre has seats arranged in rows.

Customers can reserve seats and are given a row number and seat number.

They may request bookings of several adjoining seats.

Each booking is for a particular show (i.e., the screening of a given movie at a certain time).

Shows are at an assigned date and time and scheduled in a theatre where they are screened.

The system stores the customers' telephone numbers.

Cinema booking system
- Stores seat bookings
- Stores telephone number

Theatre
- Has seats

Seat
- Arranged in rows

Seat number

Customer
- Reserves seats
- Is given row number, seat number
- Requests seat booking

Time

Movie

Date

Telephone number

Show
- Is scheduled in theatre

Seat booking
- Is for a show

Row

Row number

Stepwise Refinement

- This process of understanding a problem is called Stepwise Refinement

- We take the problem and:
  - decompose (break-down)
  - elaborate (add an appropriate level of detail)

- However, it is an iterative process involving much re-writing

- So the last step is to revise the design
  - (revisiting any of the previous steps as necessary)
  - This will continue until we are happy that we have a working design

Part 2

Software Engineering

Documentation

- Write class comments.
- Write method comments.
- Describe the overall purpose of each.
- Documenting now ensures that:
  - The focus is on *what* rather than *how*.
  - That it does not get forgotten!

Cooperation

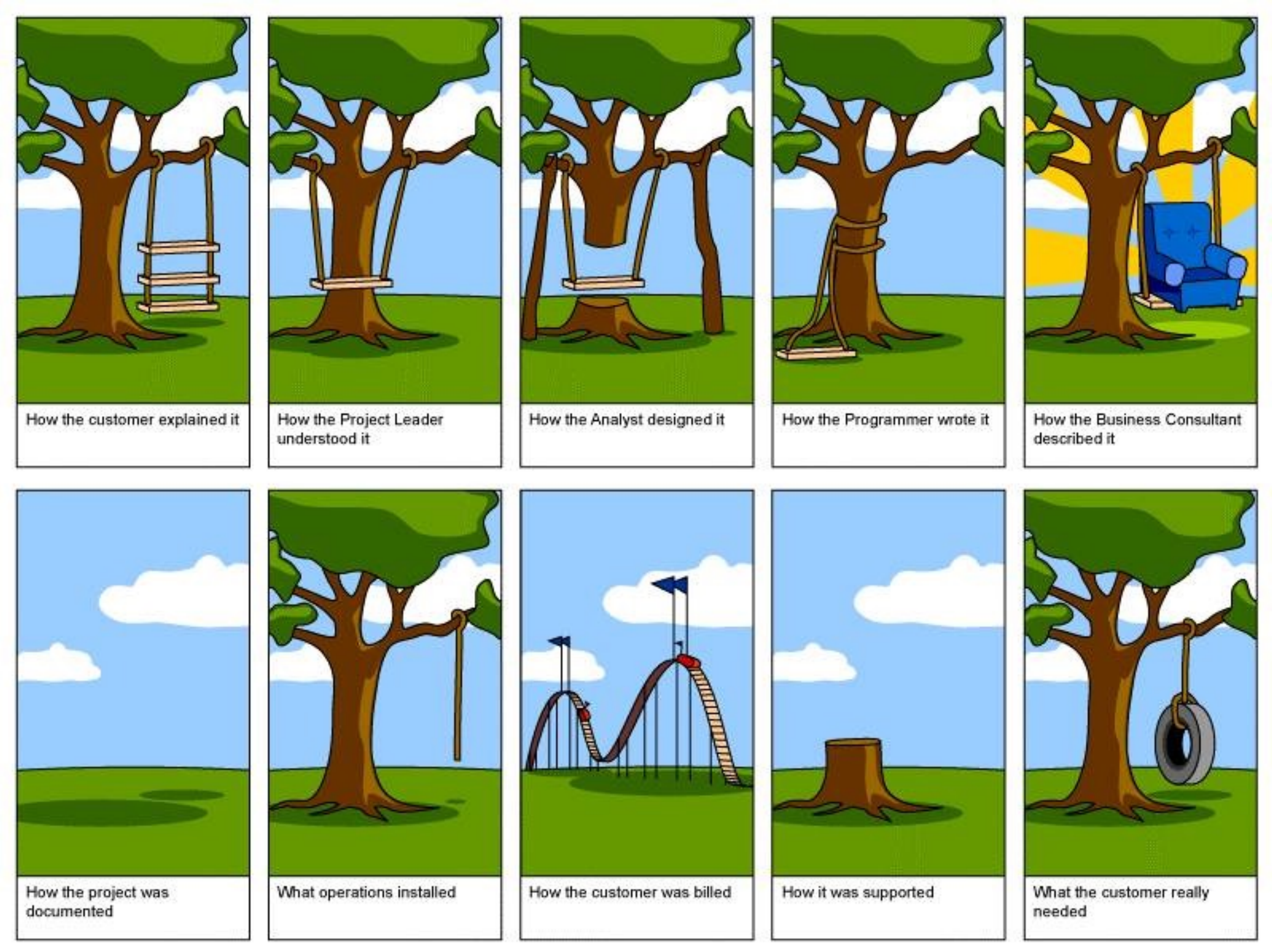
- Team-working is likely to be the norm, not the exception.
- Documentation is essential for teamworking.
- Clean O-O design, with loosely-coupled components, also supports cooperation.

Prototyping

- Supports early investigation of a system.
  - Early problem identification.
- Incomplete components can be simulated.
  - E.g. always returning a fixed result.
  - Avoid random behaviour which is difficult to reproduce.

Software Growth

- Waterfall model.
  - Analysis
  - Design
  - Implementation
  - Unit testing
  - Integration testing
  - Delivery
- No provision for iteration.


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

"How the customer explained it"
or
"The Tree Swing Story"

Iterative Development

- Use early prototyping.
- Frequent client interaction.
- Iteration over:
  - Analysis
  - Design
  - Prototype
  - Client feedback
- A growth model is the most realistic.

Part 3

Design Patterns

Using Design Patterns

- Inter-class relationships are important and can be complex.
- Some relationships recur in different applications.
- Design patterns help clarify relationships, and promote reuse.
- For example, the iterator pattern.

Pattern Structure

- A pattern name.
- The problem addressed by it.
- How it provides a solution:
  - Structures, participants, collaborations.
- Its consequences.
  - Results, trade-offs.

The Decorator Pattern

- Augments the functionality of an object.
- The decorator object wraps another object.
  - The Decorator has a similar interface.
  - Calls are relayed to the wrapped object …
  - … but the Decorator can interpolate additional actions.
- Example: `java.io.BufferedReader`
  - Wraps and augments an unbuffered **`Reader`** object.

The Singleton Pattern

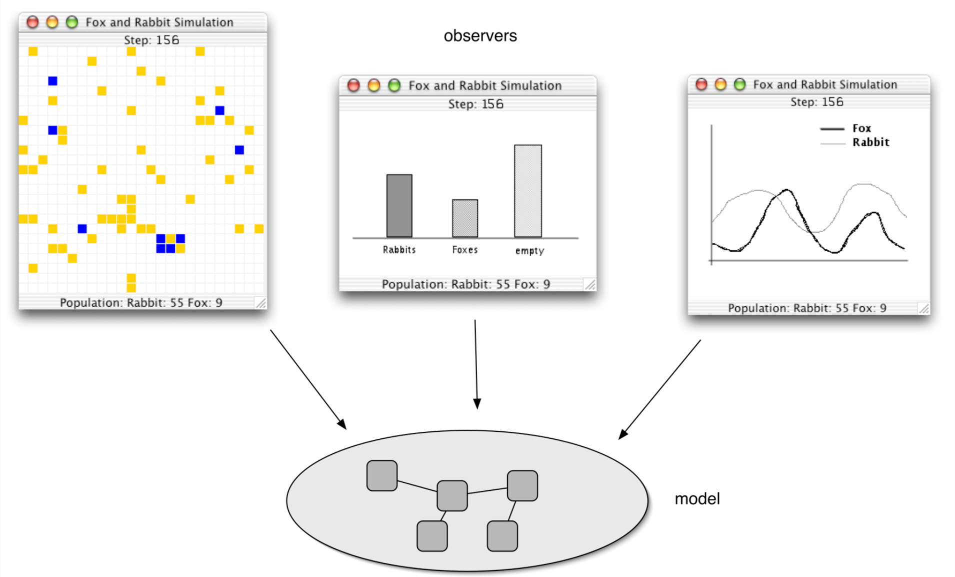
- Ensures only a single instance of a class exists.
  - All clients use the same object.
- The constructor is private to prevent external instantiation.
- Single instance obtained via a static `getInstance` method.
- Example: `Canvas` in a *GUI* project.

The Factory Method Pattern

- A creational pattern.
- Clients require an object of a particular interface type or superclass type.
- A factory method is free to return an implementing-class object or subclass object.
- The exact type returned depends on context.
- Example: `iterator` methods of the `Collection` classes.

The Observer Pattern

- Supports separation of internal model from a view of that model.
- Observer defines a one-to-many relationship between objects.
- The object-observed notifies all Observers of any state change.
- Example: Different `Views` of a database


observers
Fox and Rabbit Simulation
Step: 156
Population: Rabbit: 55 Fox: 9
Fox and Rabbit Simulation
Step: 156
Rabbits
Foxes
empty
Population: Rabbit: 55 Fox: 9
Fox and Rabbit Simulation
Step: 156
Fox
Rabbit
Population: Rabbit: 55 Fox: 9
model

Part 4

Summary

Summary

- Object Oriented Design and Analysis is complex
  - Noun Verb Analysis can get you started
  - Don’t be afraid to refactor your designs
  - There are no right answers (but some answers are better than others)

- An iterative approach to design, analysis and implementation can be beneficial.
  - Regard software systems as entities that will grow and evolve over time.

Summary

- Work in a way that facilitates collaboration with others.
- Design flexible, extendible class structures.
  - Being aware of existing design patterns will help you to do this.
- Continue to learn from your own and others' experiences.

Final Word

- Programming is both challenging and rewarding
  - It is a craft (both an art and a skill)
- Take pleasure in doing it well
- And **Have Fun!**

University of Southampton

YOUR QUESTIONS

Analysis and Design

- Noun Verb Analysis

Software Engineering

Design Patterns

- Decorator
- Singleton
- Factory method
- Observer