# SWEN30006
# Software Design and Modelling

## Subject Overview

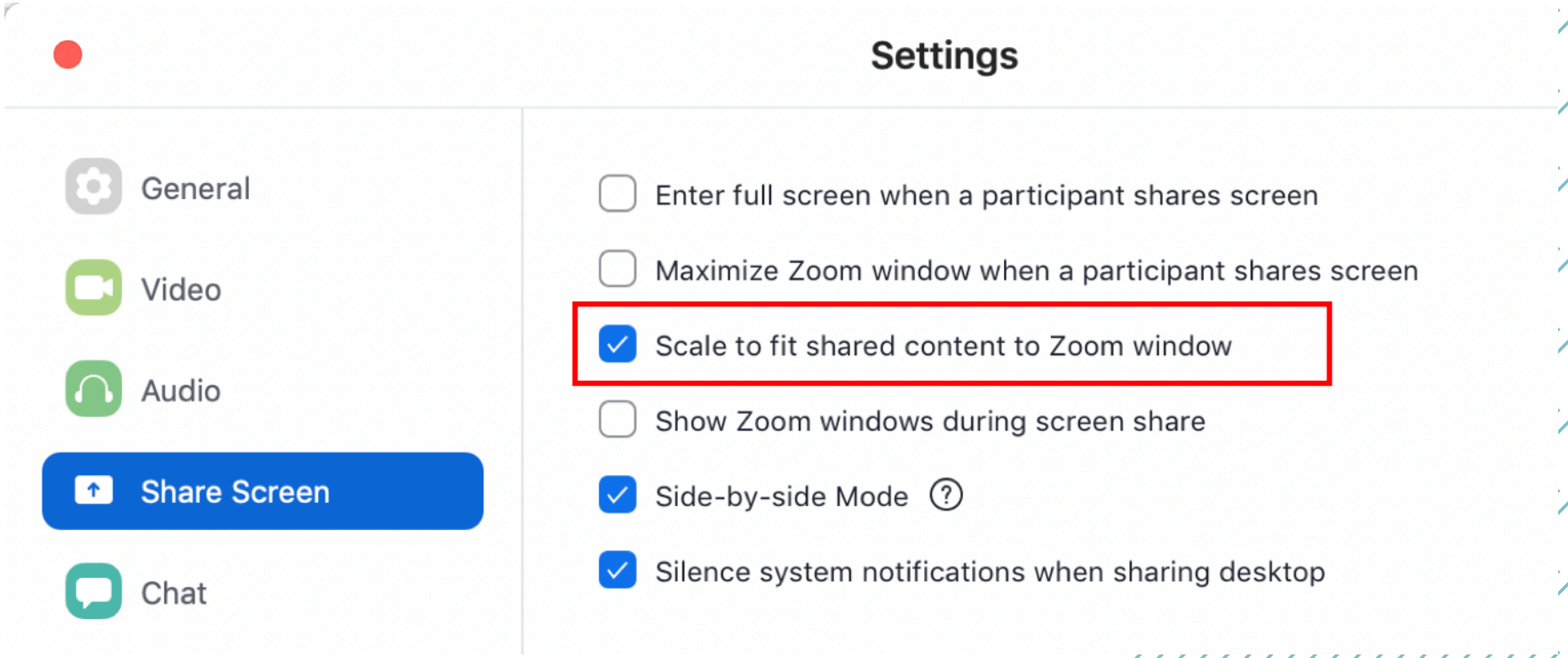
**Semester 2, 2020**

*"An ugly system is one in which there are special interfaces for everything you want to do.  Unix is the opposite. It gives you the building blocks that are sufficient for doing everything. That's what having a clean design is all about."*

*— Linus Torvalds, Just for Fun: The Story of an Accidental Revolutionary*

# If you see this slide partially, set the view option to

# “Fit to window”



# Ask your questions about the lecture

**Top**

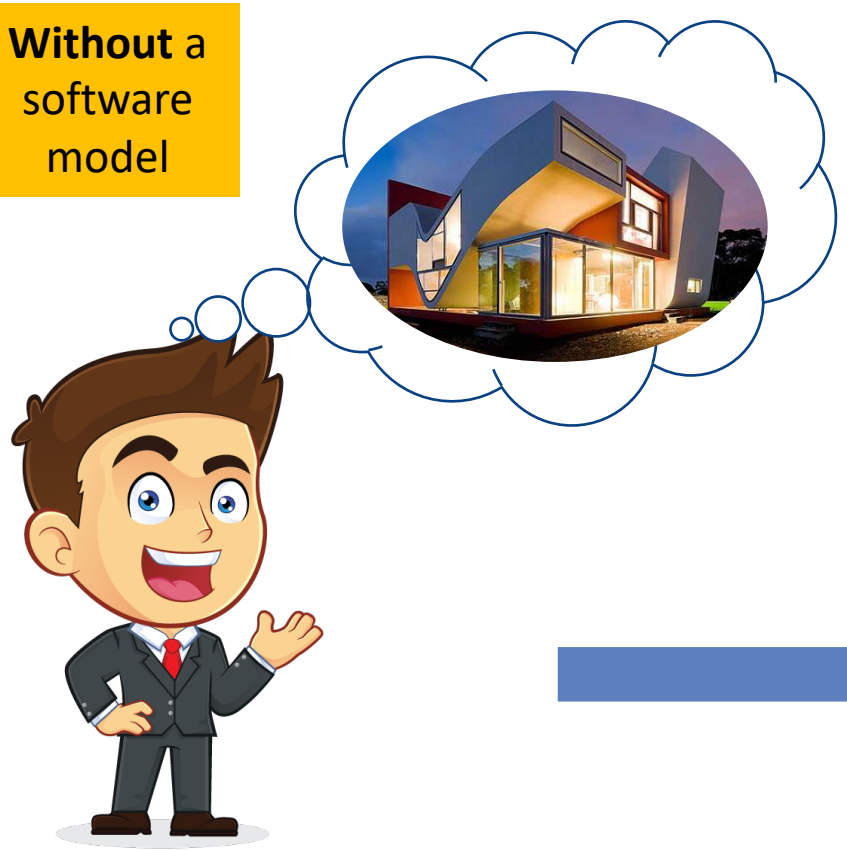
# Aims & Objective

- The **aim** of the subject is to teach you about ‘Software Modelling’ and ‘Software Design’.
- Software Design is all about *purposefully* choosing the **behaviour** and **structure** and of your software system.
  - System behaviour: how your systems responds to inputs and events
  - System structure:  how parts of the system collaborate to achieve the goals of the system
- Software Modelling is the creation of tangible, but abstract,  representations of a system so that you can communicate your design ideas, critique them and explore viable alternatives

# Why software modelling and design is important?

- A software **model** helps communication. It can be used as a shared artifact to share understanding about the system.

**Without** a software model



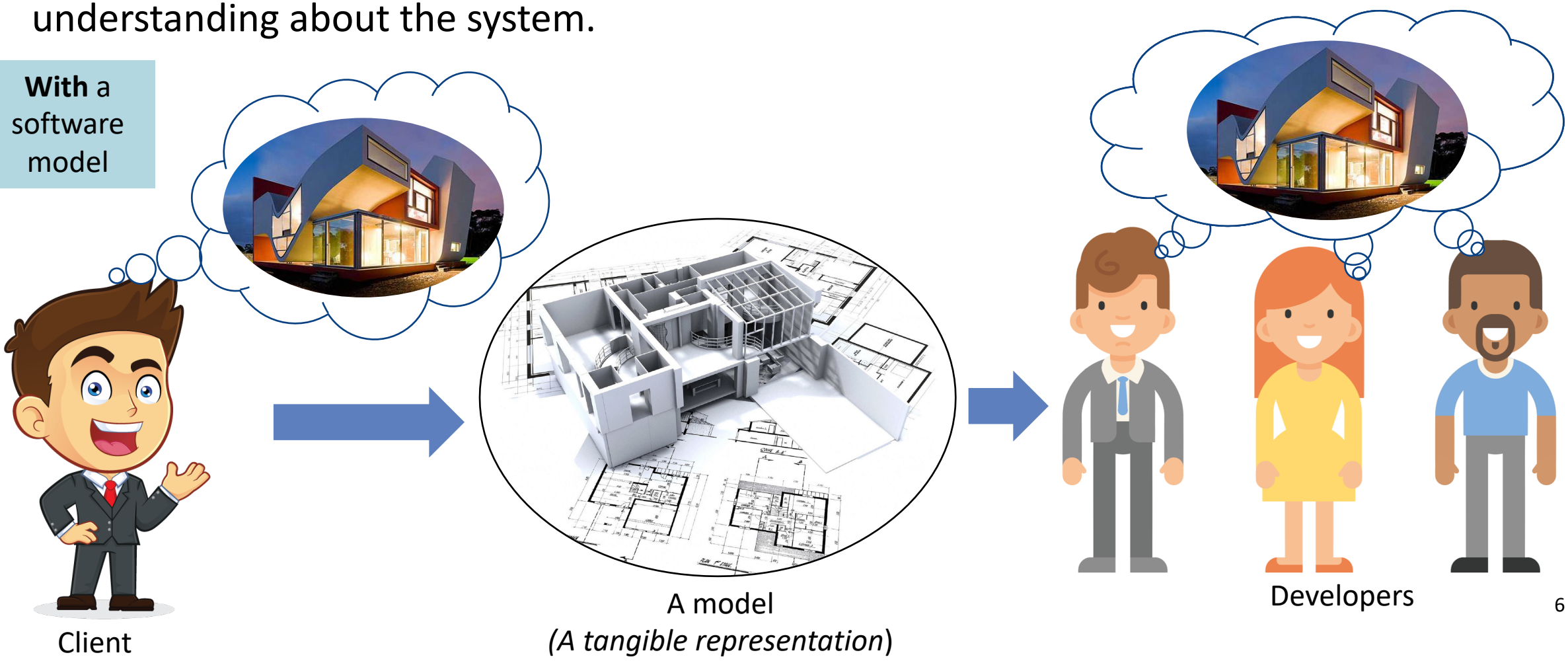
Client

Developers

# Why software modelling and design is important?

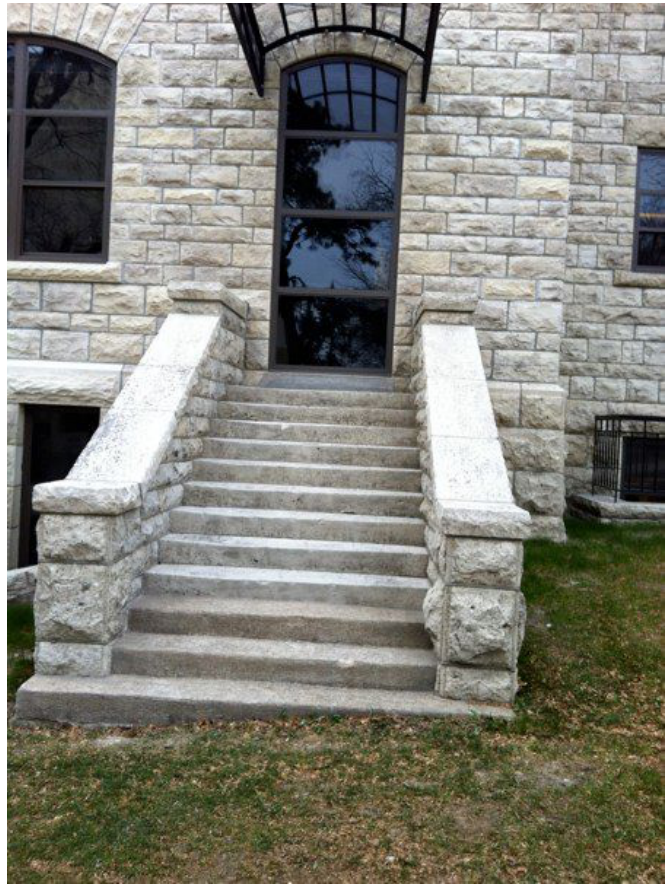
- A software **model** helps communication. It can be used as a shared artifact to share understanding about the system.

**With** a software model



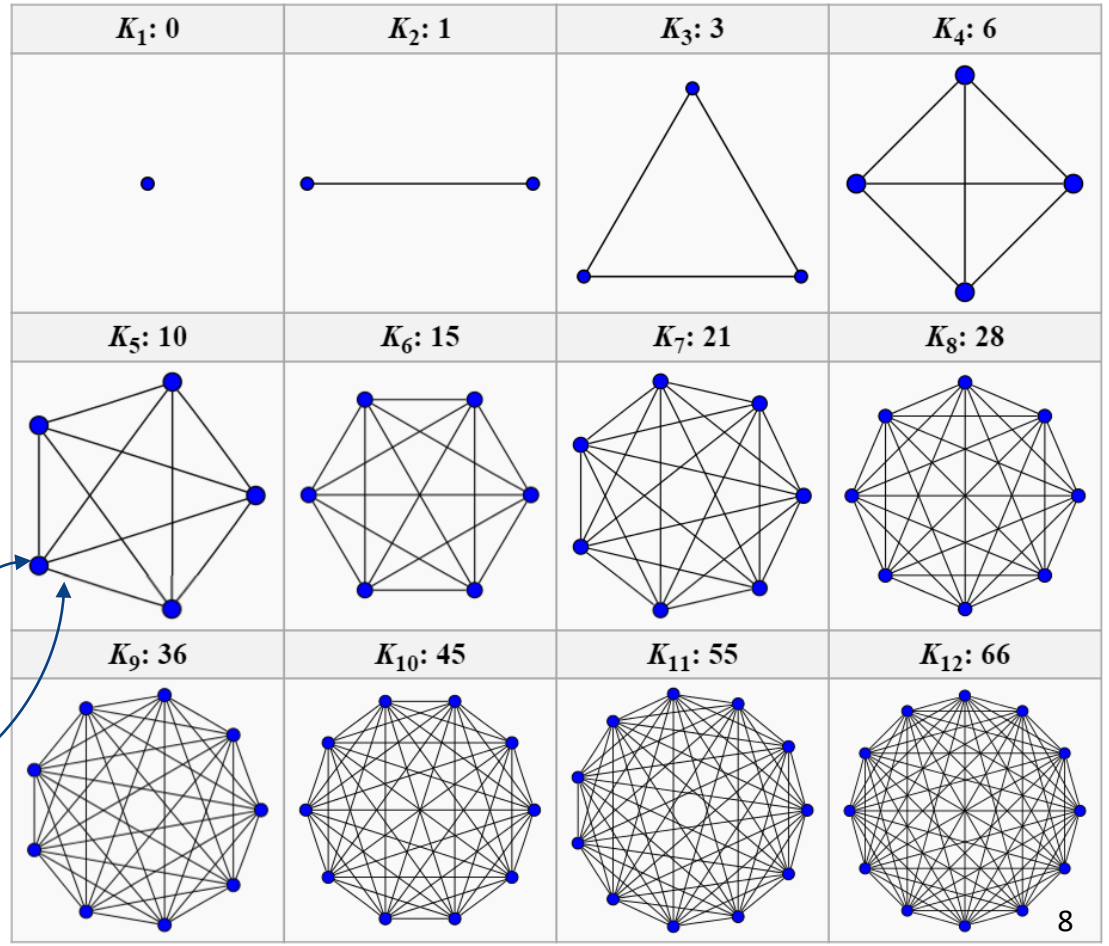
# Why software modelling and design is important?

- Software **design** should improve simplicity and effectiveness of a software system as well as ease its matainability



# Why software modelling and design is important?

- Software **design** should improve simplicity and effectiveness of a software system as well as ease its matainability
  - Software is abstract: relationships are not limited by physical proximity.
  - With some design:
    - All *N* components are interacting with each other
    - The complexity will be $K_N$: N*(N – 1)/2



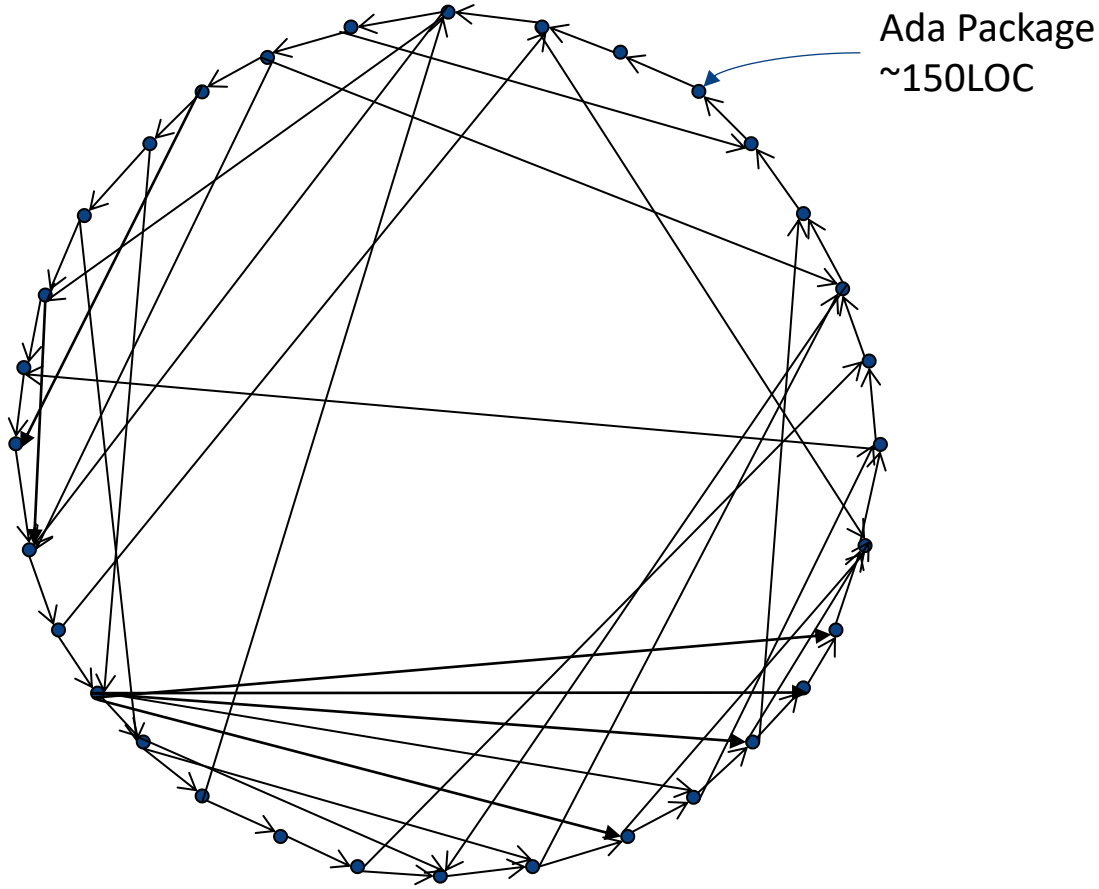

https://en.wikipedia.org/wiki/Complete_graph

# Example: Military Helicopter On-board Software

The software includes

- Navigation System
- Weapons Control System
- Communications System
- …
- Flight Control System (500 KLOC)
  - Build1 (150 KLOC)
    - Subsystem (15 KLOC - 10% of Build 1, 100 Ada packages)

Each package in the cluster depends on every other package in the cluster.
**High coupling → hard to understand → poor design**




Flight Control System
(Subsystem Build 1)

# Lessons from from the case study

- Software complexity is a problem in practice, not just in principle
- Software developers tend to add to complexity unnecessarily if unconstrained
- Requiring developers to think through the implications of a structure/design change is important in maintaining a good design
- These issues are particularly critical in large projects

# Ask your questions about the lecture

**Top**

# Software Modelling and Design subject

- The subject will focus on the object-oriented design method, with object-oriented modelling and modelling heuristics.
- UML (Unified Modelling Language) will be used as the primary modelling notation.
- Java will serve as the programming language to explore and validate the important design ideas.

**Assumed Knowledge**

- Object-Oriented Programming
- Algorithms and Data Structure
- Java Programming

**Relationship with Other Subjects**

- SWEN90009 Software Requirement and Analysis
- SWEN90007 Software Design and Architecture
- SWEN40004 Modelling Complex Software Systems

# Teaching & Learning

- **Lecture**
  - One 1-hour lecture per week
- **Workshop**
  - One 2-hours workshop per week
    *Attendance is compulsory*
  - Work on exercises as a team
  - Exercises will correspond with the lecture content of the previous week
    - Workshop 1 on Week 2 will be related to the lecture content in Week 1

## Assessment

- **8%** for Workshop **active** participation (Individual marks)
- **32%** for 2 Project assignments (Group-based marks)
- **60%** for Final examination
  - A 2-hour end-of-semester written examination

## Hurdle

To pass the subject, students must obtain at least 50% overall, 20/40 in project + workshop, and 30/60 in the final exam

*All the information and materials is available in the LMS*

# Teaching Team

**Dr Patanamon Thongtanunam**
(Subject Coordinator)
patanamon.t@unimelb.edu.au

**Dr Peter Eze**
(Lecturer)
peter.eze@unimelb.edu.au

**Peter Shi**
(Workshop: Mon 12pm & Wed 12pm)

**Green Vo**
(Workshop: Tue 12pm & Thu 12pm)

**Audrey Ahmer**
(Workshop: Tue 5pm)

**Max Plumley**
(Workshop: Wed 5pm & Thu 5pm)

**Contact details** is available at the Subject Information module > Teaching team & Contact Hours

# Case Study 1: Point-of-Sale System

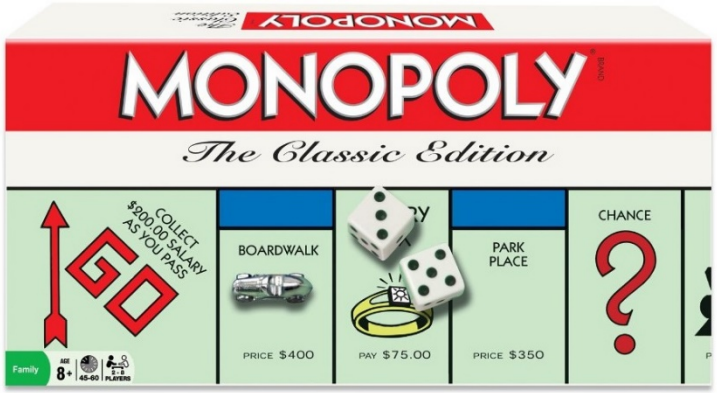
***Point-Of-Sale (POS)*** is a system to record sales and handle payments.

Includes hardware such as a register with bar code scanner and credit card reader, as well as software.

Features include:

- Interfaces to service applications, e.g. tax calculator, inventory control
- Fault-tolerant: at least capture sales and handle cash payments
- Flexibility in client-side terminals and interfaces
- Able to support different clients with different business rules, e.g. discounting policies

# Case Study 2: Monopoly Game System




A Software Simulation of Monopoly

User starts off game and watches the activities of the simulated players.

# Lecture Identification

**Lecturer: Patanamon Thongtanunam**

**Semester 2, 2020**



**These slides include materials from:**

*Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, Third Edition, by Craig Larman, Pearson Education Inc., 2005.