SWEN30006
Software Modelling and Design

SUMMARY AND EXAM

Larman Chapters: Most

Using Pen to Draw UML Diagrams in Exam is Sign of Genius or Fool.

—Philip Dart

Aims and Objectives

- The aim of the subject is to teach you about ‘Software Modelling’ and ‘Software Design’.
- Software Design is all about *purposefully* choosing the structure and behaviour of your software system.
  - The behaviour is all about how your systems responds to inputs and events, and choosing 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

Software Modelling and Design

- 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.

Larman: Required Textbook

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

- You need access to the textbook.
- We will be following the textbook closely.
- Additional references will be provided on the LMS.
- *Textbook is available in electronic form from the LMS*

Which Parts of Larman?

Look on the LMS

Which Patterns?

GRASP:

- Creator
- Information Expert
- Low Coupling
- Controller
- High Cohesion
- Polymorphism
- Indirection
- Pure Fabrication
- Protected Variations

GoF:

- Adapter
- Factory (not GoF version)
- Singleton
- Strategy
- Composite
- Façade
- Observer

UML Modelling: Which Diagrams?

- Use Case Diagrams
- Class Diagrams
- Interaction Diagrams
  - Sequence Diagrams
  - Communication Diagrams
- State Machine Diagrams
- ~~Component Diagrams~~
- ~~Activity Diagrams~~

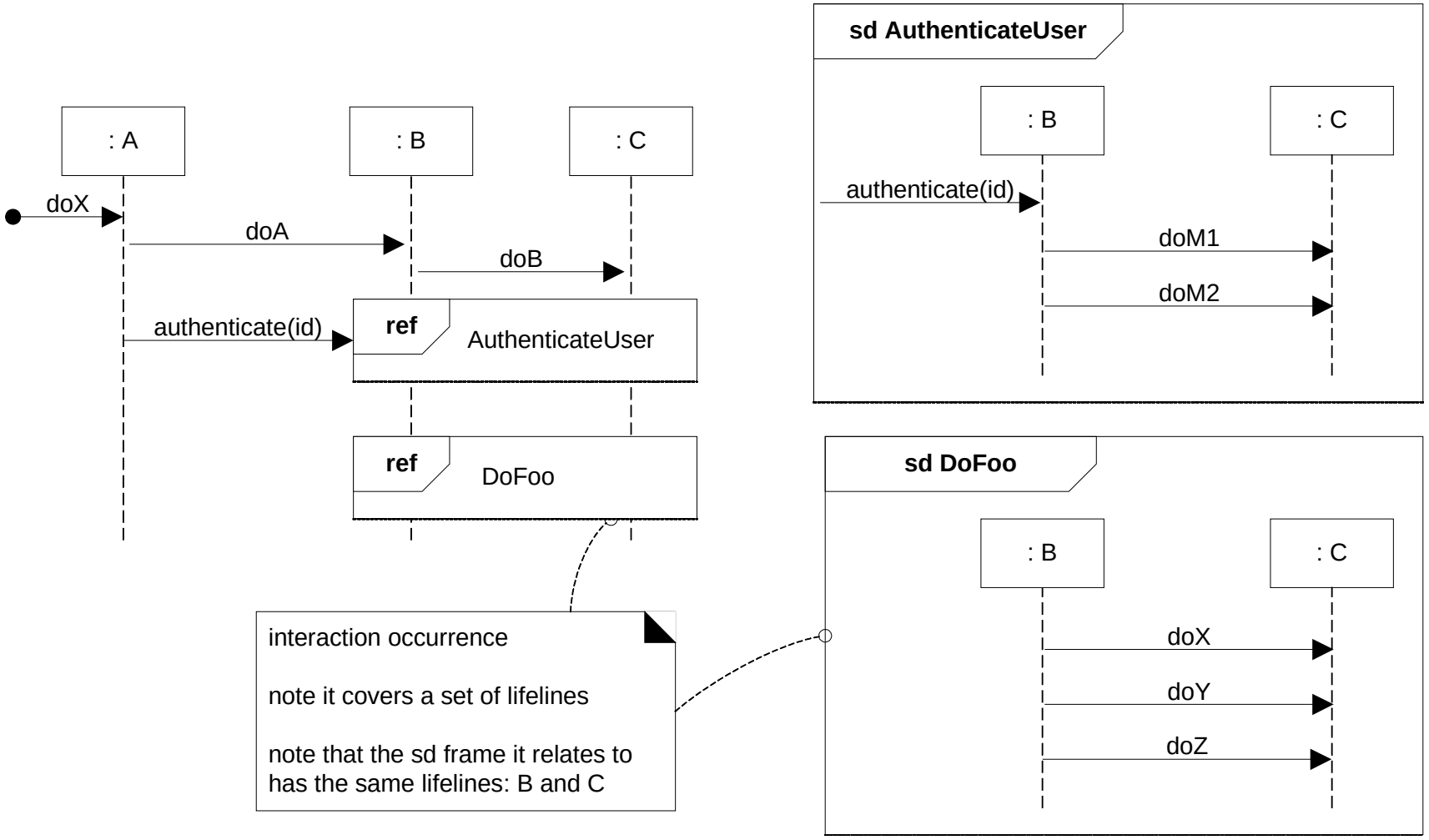
Problem vs. Solution Modelling

- Problem Space
  - Use Cases and Use Case Diagrams
  - Domain Class Diagrams
  - System Sequence Diagrams
  - *Others*
- Solution Space
  - Design Class Diagrams
  - *Others*

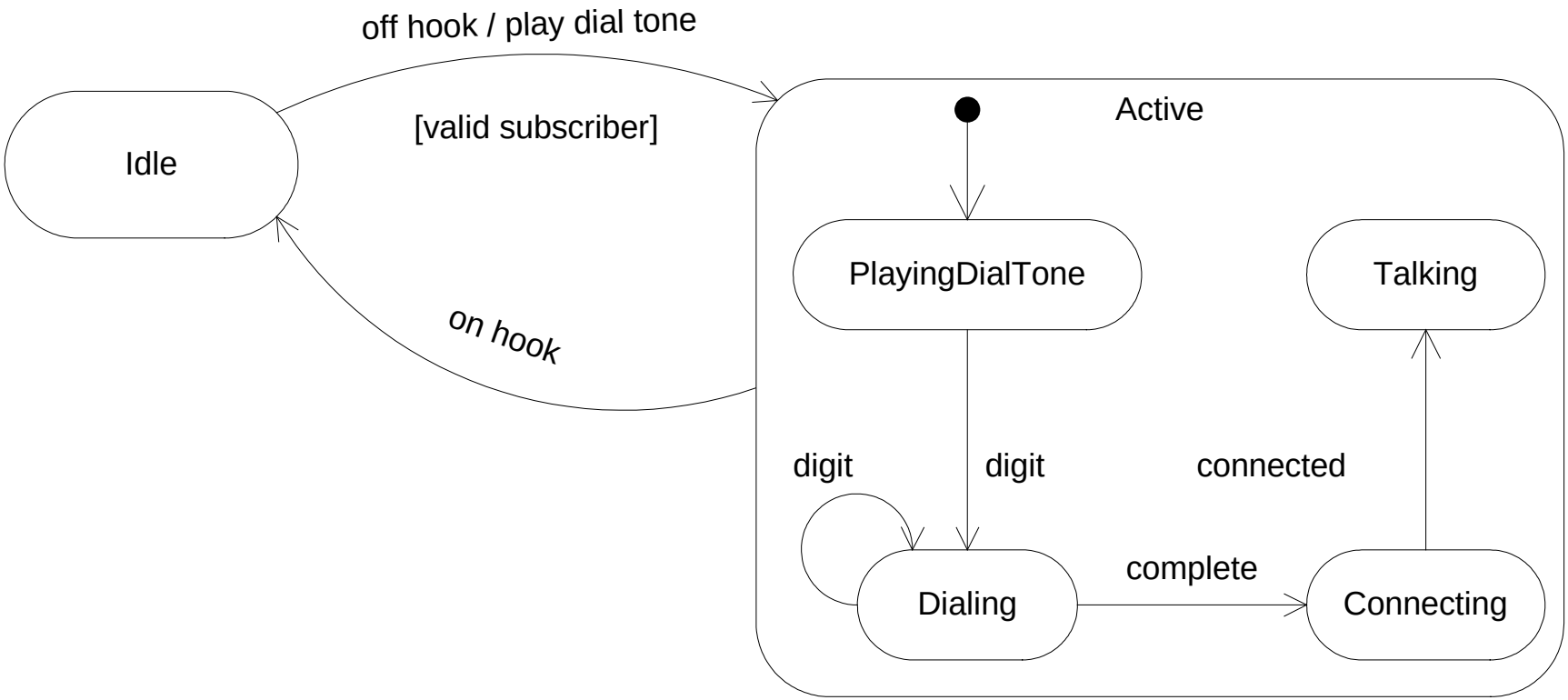
Drawing Diagrams in the Exam

- Use a pencil (and have a rubber/eraser handy)
- Know your UML notation
  - If you don't, provide a key
- Give yourself plenty of space, i.e. a whole page
- Write UML notes if you can't model
- Decompose using UML if useful, e.g.
  - Frames
  - Nested states

UML Frames: sd/ref (define/refer)



Nested States



- Transition into *Active* (via *off hook*) transitions into substate *PlayingDialTone*
- All substates of *Active* inherit the *onhook* transition.

Exam: read/write code?

- Yes!
- But …
- Not core: for this subject, coding demonstrates the effectiveness of design
- Only a 2 hour exam
- So, you could be asked to …
  - Recognize a pattern/principle in Java code
  - Write Java code to demonstrate understanding of how pattern/principle is applied

Other Exam Principles

- Software Process: role of modelling and design
- Understanding of Principles
- Read and answer the specific question
- Exam Limits: A4 and 2hrs
  - Project style design analysis not feasible
- Questions on Juggling?
  - Sadly, no!