SWINBURNE UNIVERSITY OF TECHNOLOGY

SWE30003 Software Architectures and Design

School of SCET

Semester 1, 2023

Unit Organisation

1

Unit Teaching Staff

- **Convenor/Lecturer**:
  Prof Jun Han
  Email: jhan@swin.edu.au
  Office: EN 507b
  Phone: (03) 9214 5732
  Consultation: by prior email appointment
- **Tutor**:
  Ms Mandeep Dhindsa
  Email: mdhindsa@swin.edu.au
  Consultation: by prior email appointment

2

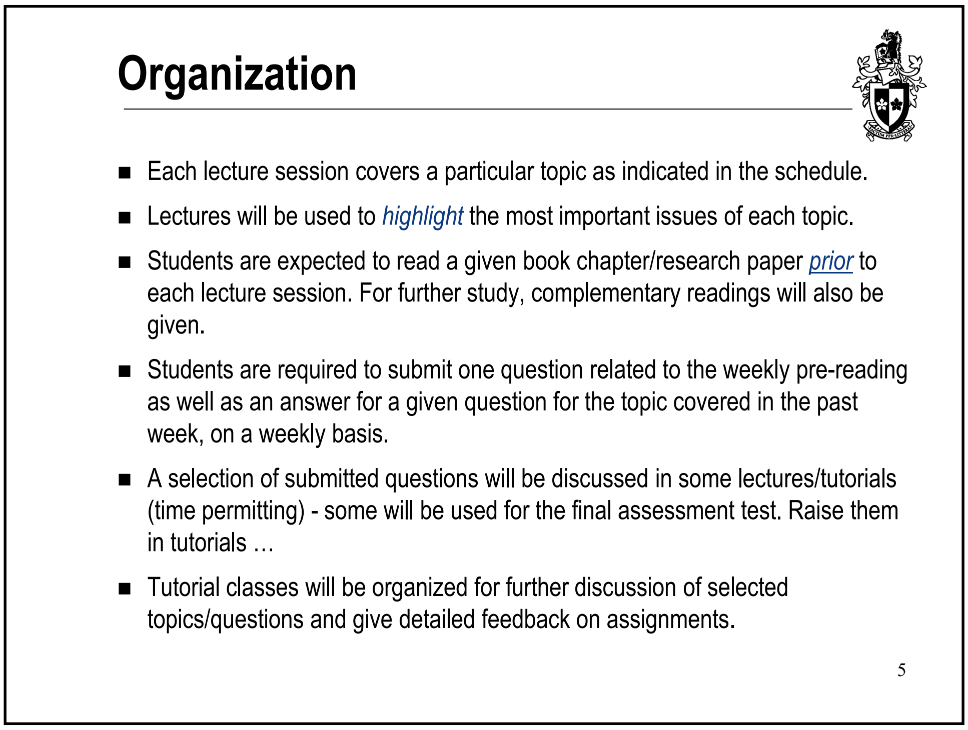
2

Aims of this Unit of Study

This unit aims to facilitate an in-depth study of state-of-the-art approaches and techniques for software system design with a special focus on the relationship between non-functional requirements and software architectures.

Goals of this Unit of Study

- To illustrate the importance of requirements analysis and appropriate notations for requirements documentation,
- To illustrate the relationship between requirements analysis and software design,
- To discuss and compare design strategies and approaches,
- To give students hands-on experience with current approaches and techniques in system design,
- To highlight the importance of verification and validation at various stages of the software development lifecycle.
- To enhance students' ability to read, understand, and discuss academic/scientific publications.

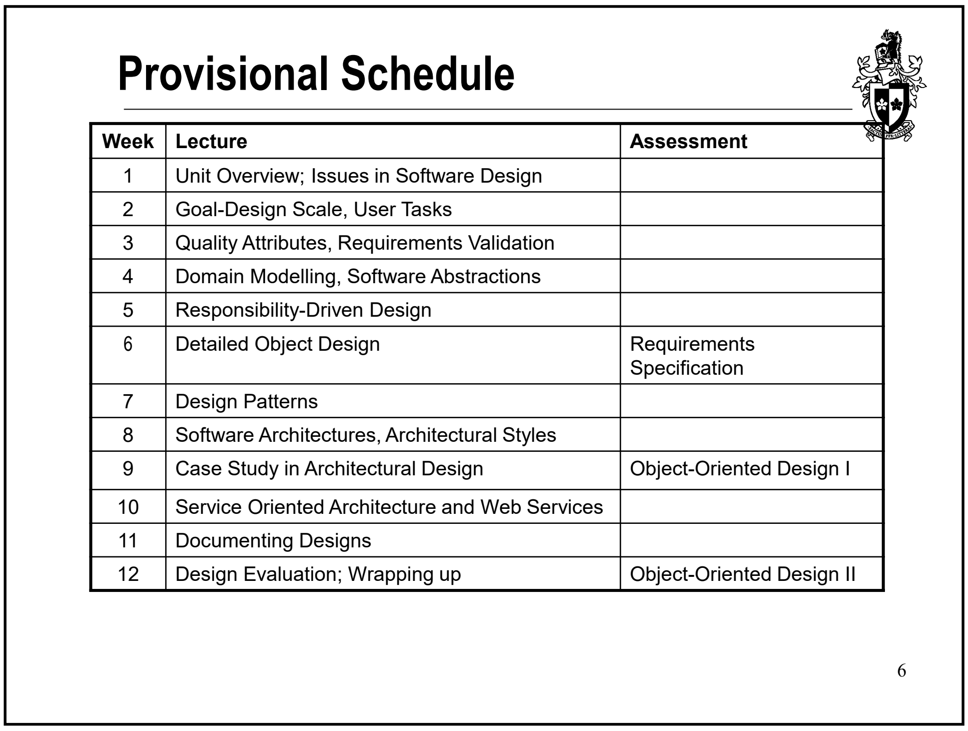


Organization

- Each lecture session covers a particular topic as indicated in the schedule.
- Lectures will be used to *highlight* the most important issues of each topic.
- Students are expected to read a given book chapter/research paper *prior* to each lecture session. For further study, complementary readings will also be given.
- Students are required to submit one question related to the weekly pre-reading as well as an answer for a given question for the topic covered in the past week, on a weekly basis.
- A selection of submitted questions will be discussed in some lectures/tutorials (time permitting) - some will be used for the final assessment test. Raise them in tutorials …
- Tutorial classes will be organized for further discussion of selected topics/questions and give detailed feedback on assignments.

5

5

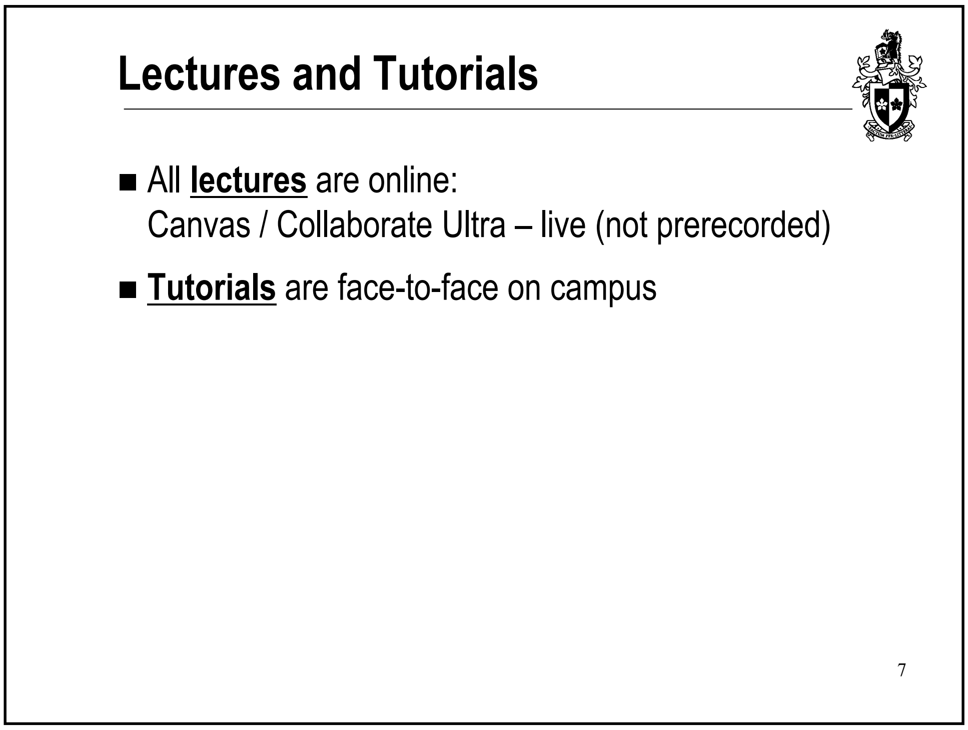


Provisional Schedule

Week	Lecture	Assessment
1	Unit Overview; Issues in Software Design	
2	Goal-Design Scale, User Tasks	
3	Quality Attributes, Requirements Validation	
4	Domain Modelling, Software Abstractions	
5	Responsibility-Driven Design	
6	Detailed Object Design	Requirements Specification
7	Design Patterns	
8	Software Architectures, Architectural Styles	
9	Case Study in Architectural Design	Object-Oriented Design I
10	Service Oriented Architecture and Web Services	
11	Documenting Designs	
12	Design Evaluation; Wrapping up	Object-Oriented Design II

6

6

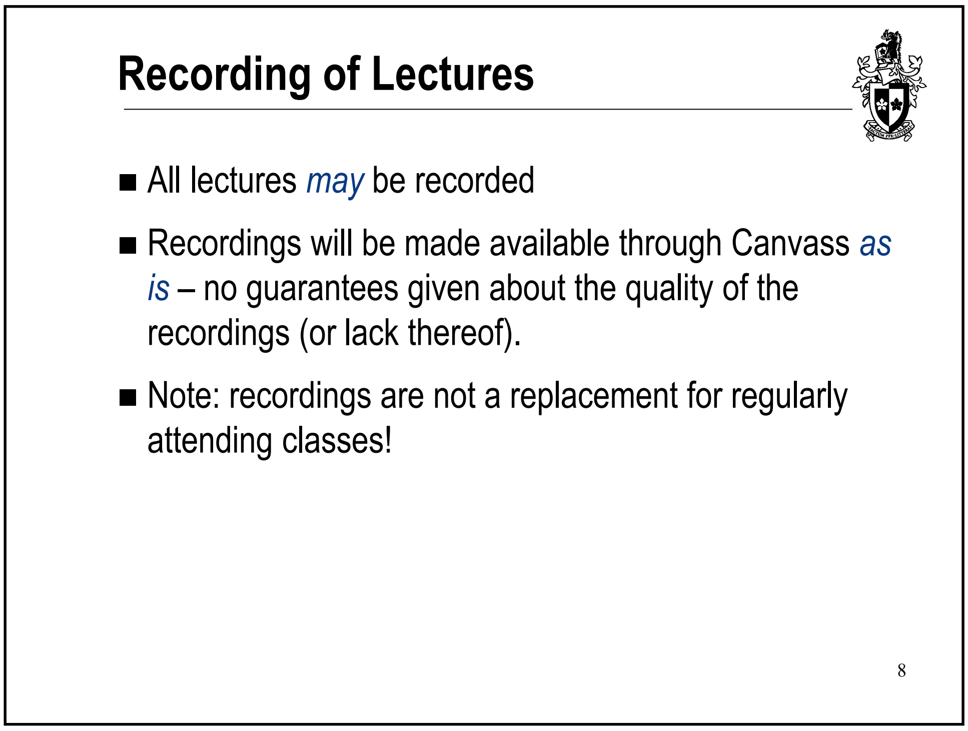


Lectures and Tutorials

- All **lectures** are online:
  Canvas / Collaborate Ultra – live (not prerecorded)
- **Tutorials** are face-to-face on campus

7

7

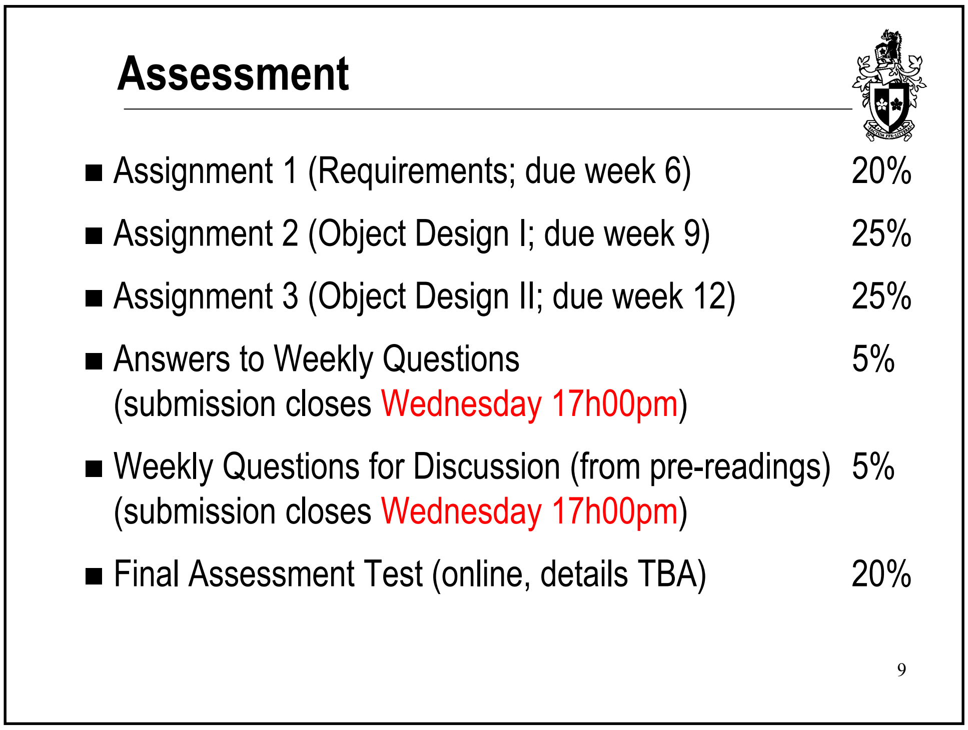


Recording of Lectures

- All lectures *may* be recorded
- Recordings will be made available through Canvass *as is* – no guarantees given about the quality of the recordings (or lack thereof).
- Note: recordings are not a replacement for regularly attending classes!

8

8

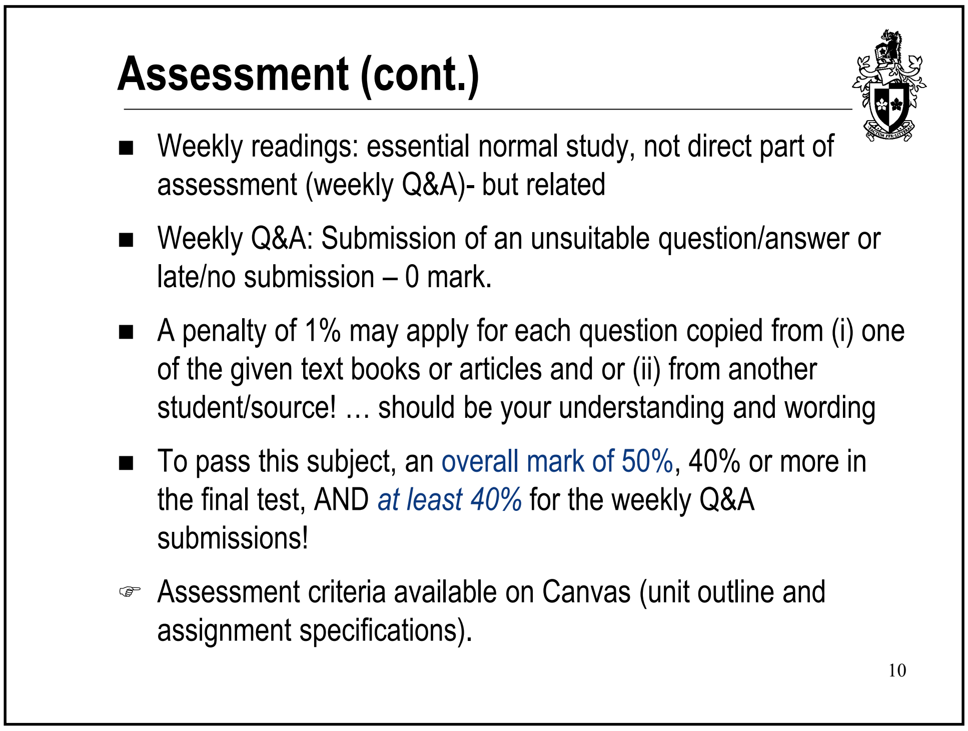


Assessment

■ Assignment 1 (Requirements; due week 6)	20%
■ Assignment 2 (Object Design I; due week 9)	25%
■ Assignment 3 (Object Design II; due week 12)	25%
■ Answers to Weekly Questions (submission closes Wednesday 17h00pm)	5%
■ Weekly Questions for Discussion (from pre-readings) (submission closes Wednesday 17h00pm)	5%
■ Final Assessment Test (online, details TBA)	20%

9

9



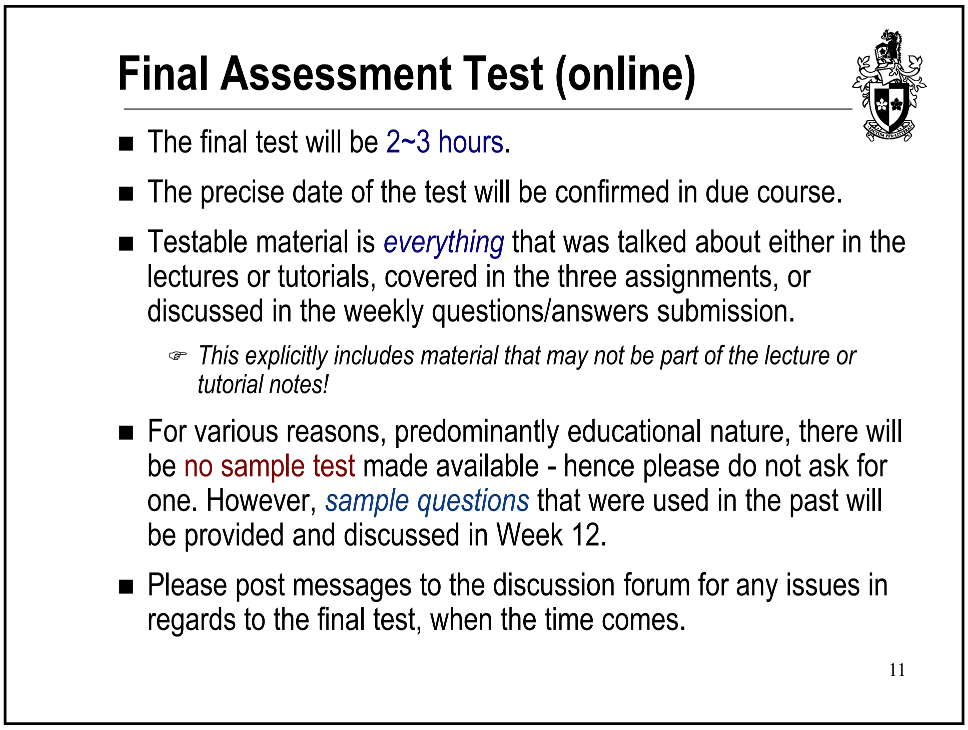
Assessment (cont.)

- Weekly readings: essential normal study, not direct part of assessment (weekly Q&A)- but related
- Weekly Q&A: Submission of an unsuitable question/answer or late/no submission – 0 mark.
- A penalty of 1% may apply for each question copied from (i) one of the given text books or articles and or (ii) from another student/source! … should be your understanding and wording
- To pass this subject, an overall mark of 50%, 40% or more in the final test, AND *at least 40%* for the weekly Q&A submissions!

☞ Assessment criteria available on Canvas (unit outline and assignment specifications).

10

10

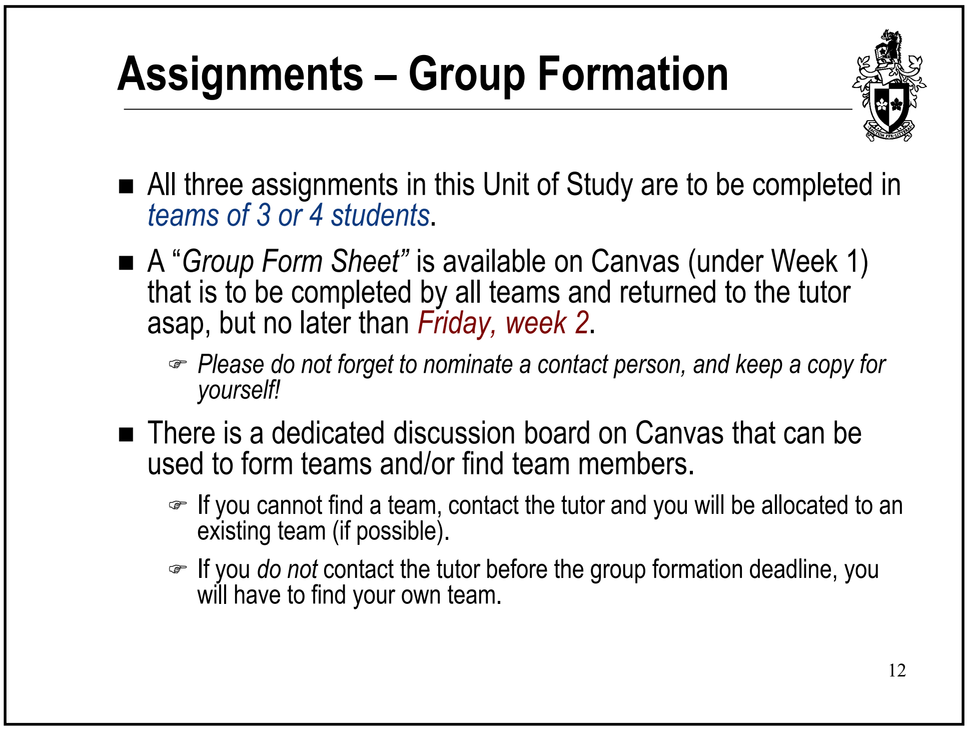


Final Assessment Test (online)

- The final test will be 2~3 hours.
- The precise date of the test will be confirmed in due course.
- Testable material is *everything* that was talked about either in the lectures or tutorials, covered in the three assignments, or discussed in the weekly questions/answers submission.
  - ☞ *This explicitly includes material that may not be part of the lecture or tutorial notes!*
- For various reasons, predominantly educational nature, there will be no sample test made available - hence please do not ask for one. However, *sample questions* that were used in the past will be provided and discussed in Week 12.
- Please post messages to the discussion forum for any issues in regards to the final test, when the time comes.

11

11

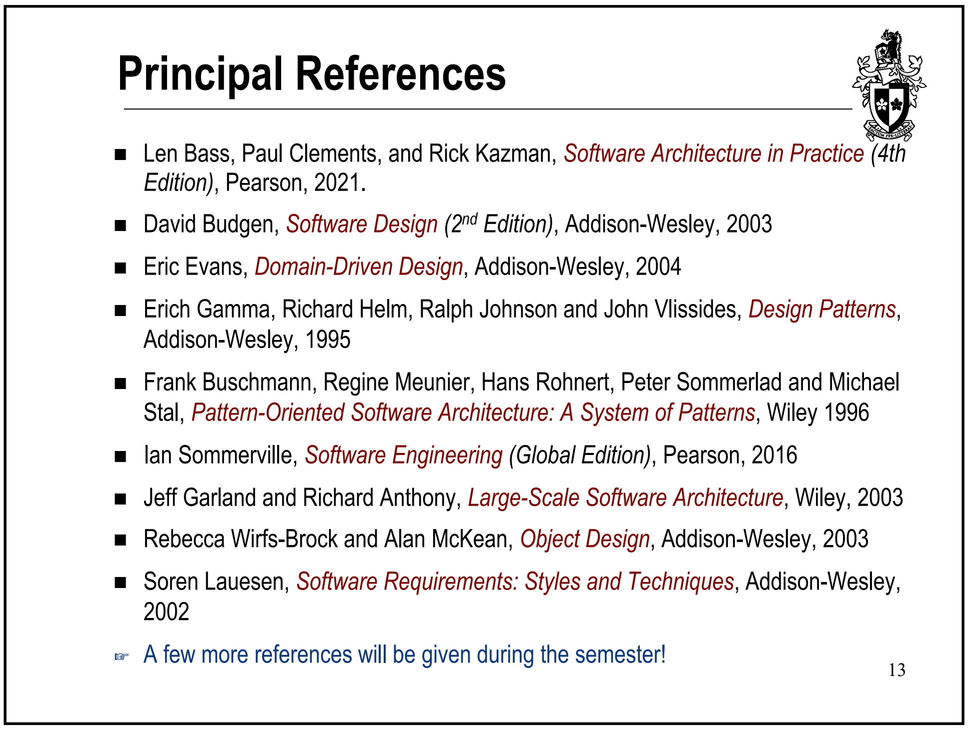


Assignments – Group Formation

- All three assignments in this Unit of Study are to be completed in *teams of 3 or 4 students*.
- A "*Group Form Sheet*" is available on Canvas (under Week 1) that is to be completed by all teams and returned to the tutor asap, but no later than *Friday, week 2*.
  - ☞ *Please do not forget to nominate a contact person, and keep a copy for yourself!*
- There is a dedicated discussion board on Canvas that can be used to form teams and/or find team members.
  - ☞ If you cannot find a team, contact the tutor and you will be allocated to an existing team (if possible).
  - ☞ If you *do not* contact the tutor before the group formation deadline, you will have to find your own team.

12

12



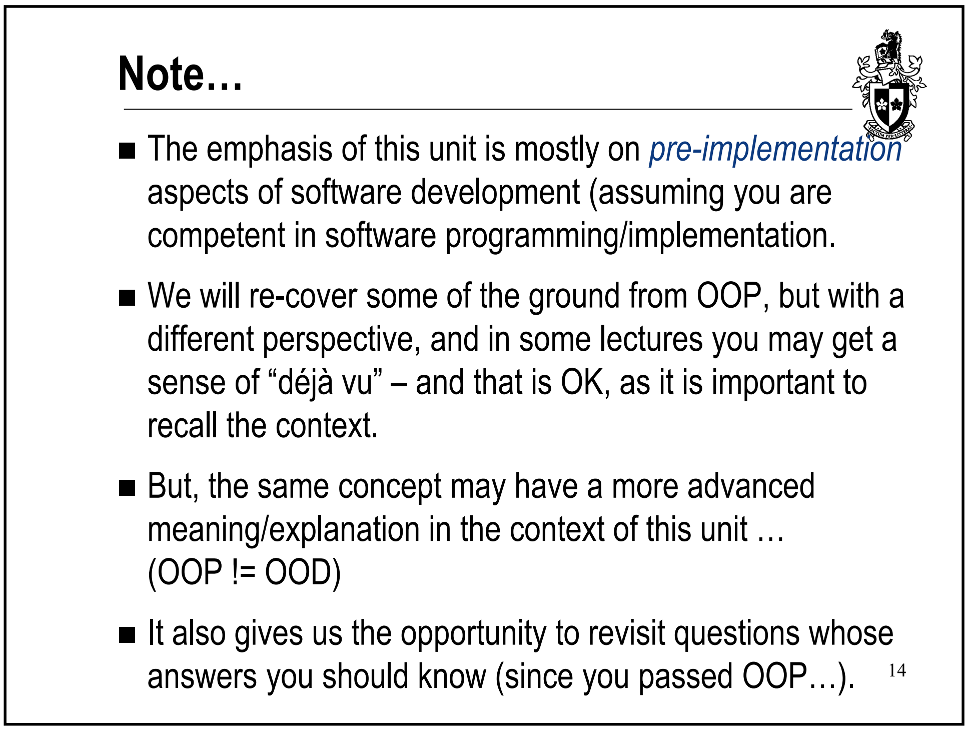
Principal References

- Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice (4th Edition)*, Pearson, 2021.
- David Budgen, *Software Design (2nd Edition)*, Addison-Wesley, 2003
- Eric Evans, *Domain-Driven Design*, Addison-Wesley, 2004
- Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, *Design Patterns*, Addison-Wesley, 1995
- Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad and Michael Stal, *Pattern-Oriented Software Architecture: A System of Patterns*, Wiley 1996
- Ian Sommerville, *Software Engineering (Global Edition)*, Pearson, 2016
- Jeff Garland and Richard Anthony, *Large-Scale Software Architecture*, Wiley, 2003
- Rebecca Wirfs-Brock and Alan McKean, *Object Design*, Addison-Wesley, 2003
- Soren Lauesen, *Software Requirements: Styles and Techniques*, Addison-Wesley, 2002

☞ A few more references will be given during the semester!

13

13

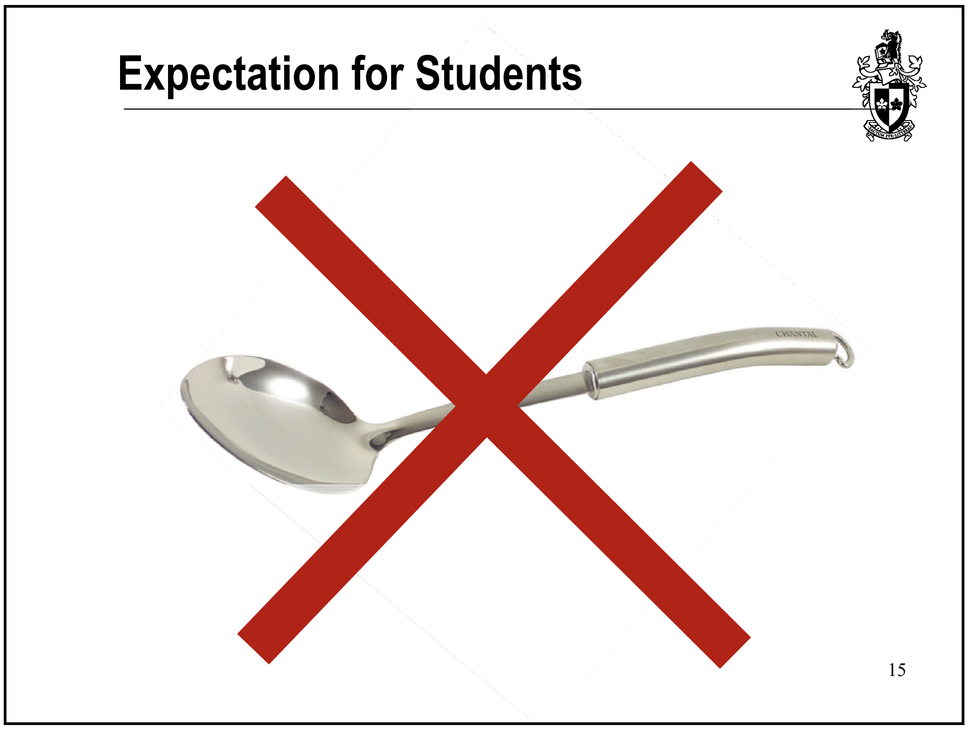


Note…

- The emphasis of this unit is mostly on *pre-implementation* aspects of software development (assuming you are competent in software programming/implementation.
- We will re-cover some of the ground from OOP, but with a different perspective, and in some lectures you may get a sense of “déjà vu” – and that is OK, as it is important to recall the context.
- But, the same concept may have a more advanced meaning/explanation in the context of this unit … (OOP != OOD)
- It also gives us the opportunity to revisit questions whose answers you should know (since you passed OOP…).

14

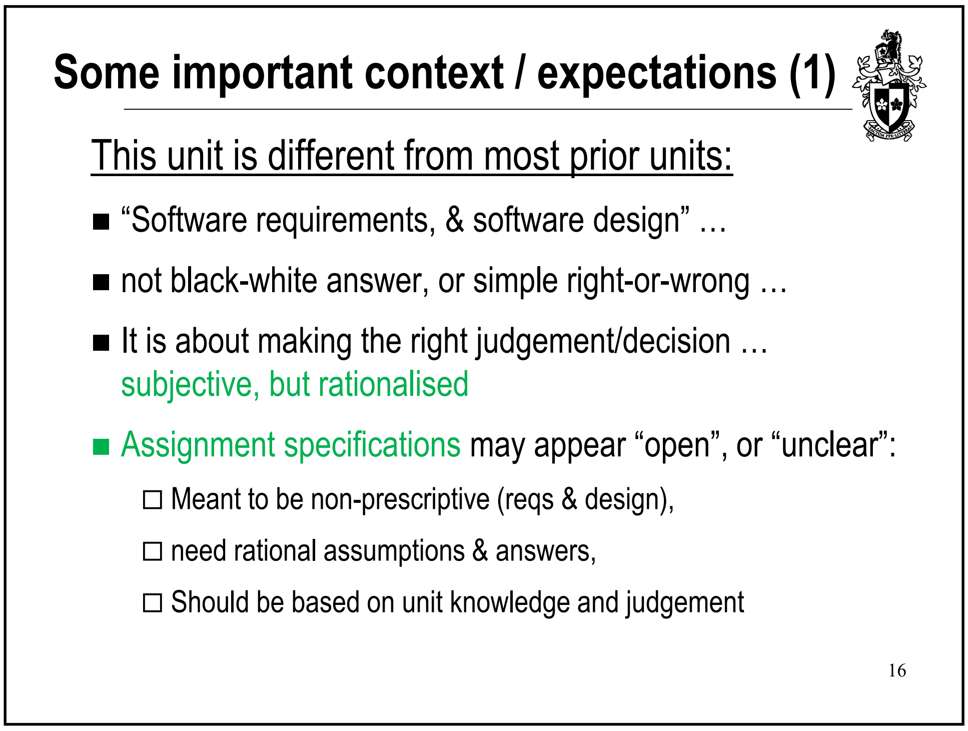
14



Expectation for Students

15

15



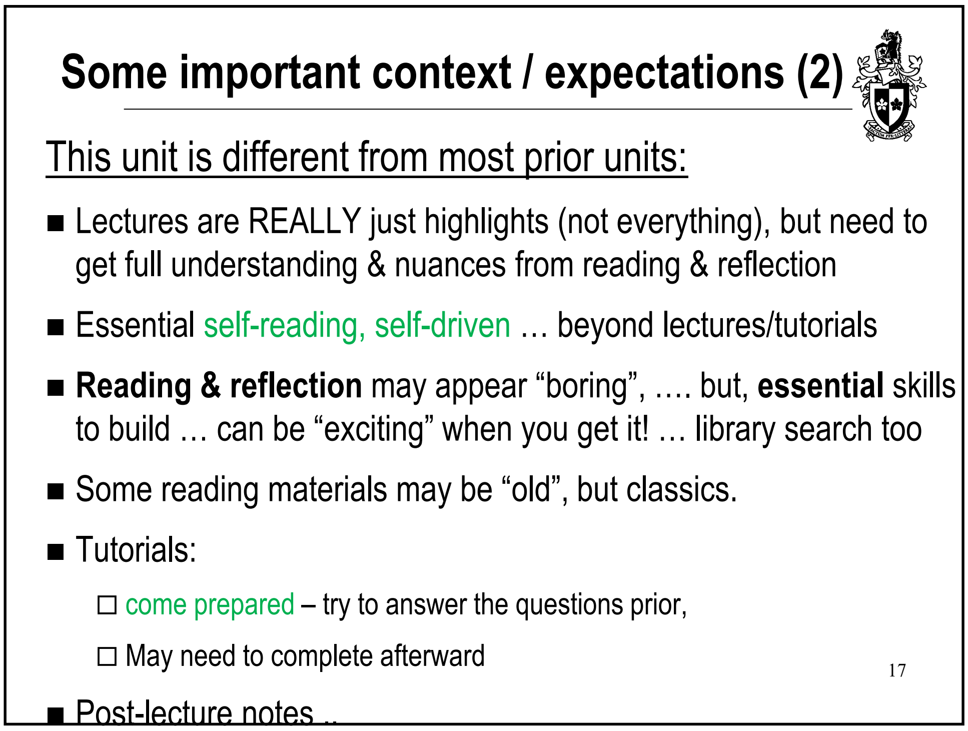
Some important context / expectations (1)

<u>This unit is different from most prior units:</u>

- "Software requirements, & software design" …
- not black-white answer, or simple right-or-wrong …
- It is about making the right judgement/decision …
  subjective, but rationalised
- Assignment specifications may appear "open", or "unclear":
  - ☐ Meant to be non-prescriptive (reqs & design),
  - ☐ need rational assumptions & answers,
  - ☐ Should be based on unit knowledge and judgement

16

16



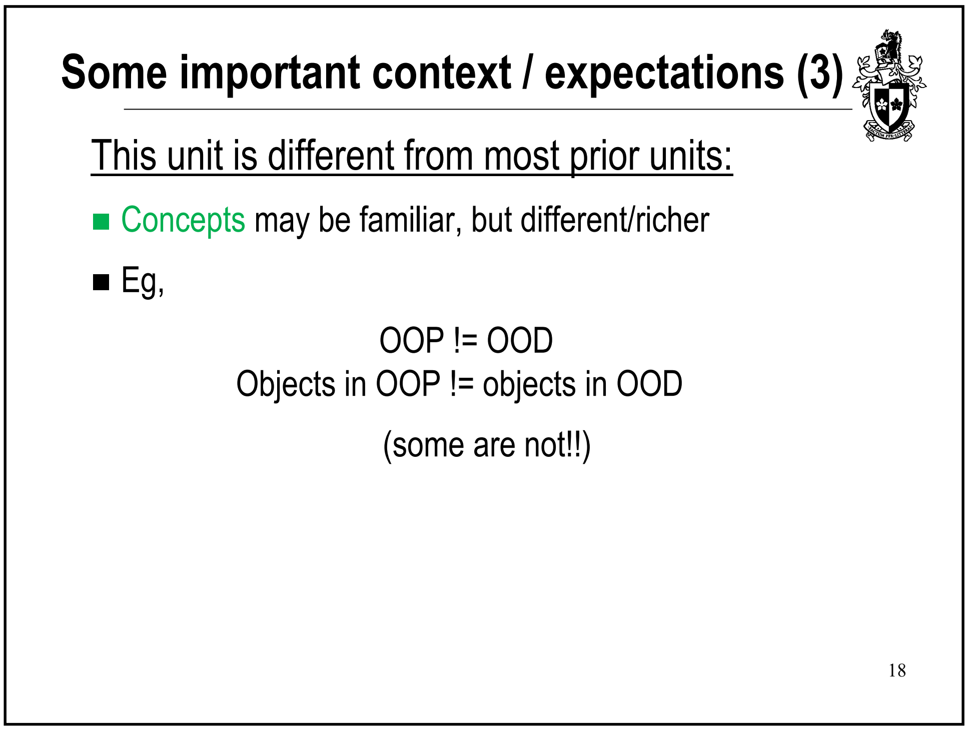
Some important context / expectations (2)

This unit is different from most prior units:

- Lectures are REALLY just highlights (not everything), but need to get full understanding & nuances from reading & reflection
- Essential self-reading, self-driven … beyond lectures/tutorials
- **Reading & reflection** may appear "boring", …. but, **essential** skills to build … can be "exciting" when you get it! … library search too
- Some reading materials may be "old", but classics.
- Tutorials:
  - ☐ come prepared – try to answer the questions prior,
  - ☐ May need to complete afterward
- Post-lecture notes ..

17

17



Some important context / expectations (3)

This unit is different from most prior units:

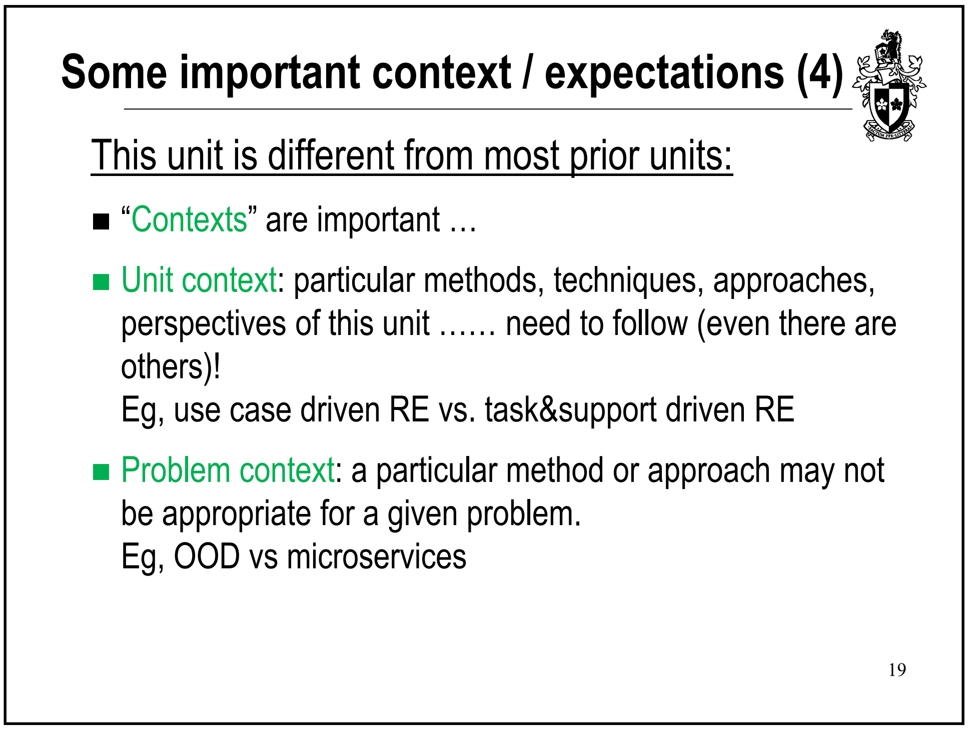
- Concepts may be familiar, but different/richer
- Eg,

OOP != OOD
Objects in OOP != objects in OOD

(some are not!!)

18

18



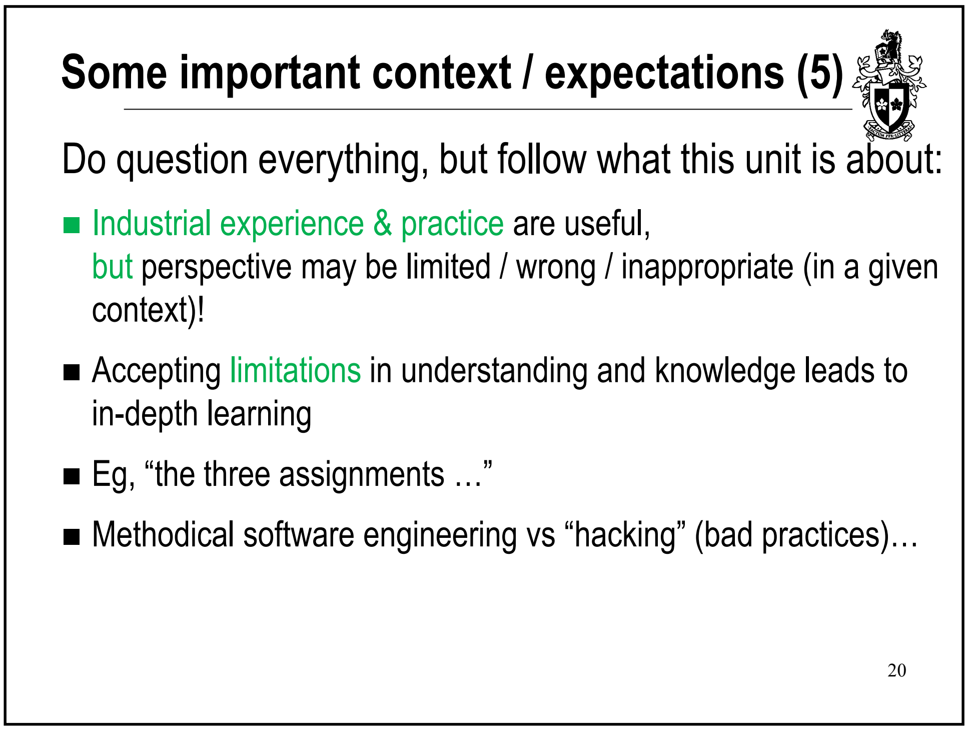
Some important context / expectations (4)

This unit is different from most prior units:

- "Contexts" are important …
- Unit context: particular methods, techniques, approaches, perspectives of this unit …… need to follow (even there are others)!
  Eg, use case driven RE vs. task&support driven RE
- Problem context: a particular method or approach may not be appropriate for a given problem.
  Eg, OOD vs microservices

19

19



Some important context / expectations (5)

Do question everything, but follow what this unit is about:

- Industrial experience & practice are useful,
  but perspective may be limited / wrong / inappropriate (in a given context)!
- Accepting limitations in understanding and knowledge leads to in-depth learning
- Eg, "the three assignments …"
- Methodical software engineering vs "hacking" (bad practices)…

20

20

Some important context / expectations (6)

True group/team work is essential:

- Industrial practice
- Collaboration leads to good outcomes
- In particular, in rationalised (design) decision-making (not just the first thought about)
- Team submission should NOT be a simple “assembly” of individual work – un-refined, inconsistent, not to mention sub-optimal

21

21

Quotes (1)

“Although i missed a few, asking questions from weekly readings really forced me to read literature from the broader context of the field. The talk about developing a language was very interesting and i'm very glad i was forced to watch it. You never know what you don't know and i would rarely consume this literature on my own time.”

22

22

Quotes (2)

"The topics. This class has been wonderfully helpful in showing how a system is designed; a step that has been severely lacking in BACS up to now. The three assignments providing the step by step experience of designing and implementing a system also makes me a lot more confident and comfortable going into my capstone units next year."

23

23

So …… you know what to do … ☺

24

24