what is offered here?

**Fundamentals & Overview**
as well as perspectives, paths, methods, implementations, and open questions

**Concurrent & Distributed Systems**

who could be interested in this?

anybody who …

… wants to work with **real-world scale** computer systems

… would like to learn how to

analyse and design operational and robust systems

… would like to understand more about the existing trade-off between

theory, the real-world, traditions, and pragmatism in computer science

… would like to understand why concurrent systems are

an **essential basis** for most contemporary devices and systems
**Organization & Contents**

**who are these people? – introductions**

This course will be given by

Uwe R. Zimmer & Alistair Rendell

Your individual tutors are

Abigail Thomas, Benjamin Wang,
Liyang (Leon) Guan, Michael Bennett,
Migara Liyanagamage,
Nathan Yong,
Patrick Chieppe

---

**Organization & Contents**

**Text book for the course**

[Ben-Ari06]

M. Ben-Ari

*Principles of Concurrent and Distributed Programming*


Many algorithms and concepts for the course are in there

– but not all!

References for specific aspects of the course are provided during the course and are found on our web-site.

---

**Organization & Contents**

**how will this all be done?**

- **Lectures:**
  - 2 x 1.5 hours lectures per week ... all the nice stuff
    - Monday 16:00, Thursday 17:00 (both in R.N. Robertson - which is: here)

- **Laboratories:**
  - 2 hours per week ... all the rough stuff
  - time slots: on our web-site – all in CSIT Nxxx laboratories
  - enrolment: https://cs.anu.edu.au/streams/ (will open after first lecture)

- **Resources:**
  - Introduced in the lectures and collected on the course page:
    - https://cs.anu.edu.au/courses/comp2310/... as well as schedules, slides, sources, links to forums, etc. pp. ... keep an eye on this page!

- **Assessment:**
  - Exam at the end of the course (55%)
    - plus one hurdle lab in week 4 (5%)
    - plus two assignments (15% + 15%)
    - plus one mid-semester exam (10%)

---

**Organization & Contents**

**Topics**

1. Concurrency [3]
5. Scheduling [2]
7. Architectures for CDS [1]
8. Distributed systems [7]
Organization & Contents

Topics

1. Concurrency [3]
   1.1. Forms of concurrency [1]
       • Coupled dynamical systems
   1.2. Models and terminology [1]
       • Abstractions
       • Interleaving
       • Atomicity
       • Proofs in concurrent and distributed systems
   1.3. Processes & threads [1]
       • Basic definitions
       • Process states
       • Implementations

   2.1. by shared variables [1]
       • Failure possibilities
       • Dekker’s algorithm
   2.2. by test-and-set hardware support [0.5]
       • Minimal hardware support
   2.3. by semaphores [0.5]
       • Dijkstra definition
       • OS semaphores

   3.1. Shared memory synchronization [2]
       • Semaphores
       • Cond. variables
       • Conditional critical regions
       • Monitors
       • Protected objects
       • Asynchronous / synchronous
       • Remote invocation / rendezvous
       • Message structure
       • Addressing

   4.1. Correctness under non-determinism [1]
       • Forms of non-determinism
       • Non-determinism in concurrent/ distributed systems
       • Is consistency/correctness plus non-determinism a contradiction?
   4.2. Select statements [1]
       • Forms of non-deterministic message reception

5. Scheduling [2]


7. Architectures for CDS [1]

8. Distributed systems [7]
Topics

1. Concurrency [3]
5. Scheduling [2]
7. Architectures for CDS [1]
8. Distributed systems [7]

7. Architectures for CDS [1]

8. Distributed systems [7]
1. Concurrency [3]
   1.1. Forms of concurrency [1]
   - Coupled dynamical systems
   - Interleaving
   - Atomicity
   - Proofs in concurrent and distributed systems
   1.2. Models and terminology [1]
   - Abstractions
   - Interleaving
   - Atomicity
   - Proofs in concurrent and distributed systems
   1.3. Processes & threads [1]
   - Basic definitions
   - Process states
   - Implementations

   2.1. by shared variables [1]
   - Failure possibilities
   - Dekker’s algorithm
   2.2. by test-and-set hardware support [0.5]
   - Minimal hardware support
   2.3. by semaphores [0.5]
   - Dijkstra definition
   - OS semaphores

   3.1. Shared memory synchronization [2]
   - Semaphores
   - Cond. variables
   - Conditional critical regions
   - Monitors
   - Protected objects
   - Asynchronous / synchronous
   - Remote invocation / rendezvous
   - Message structure
   - Addressing

   4.1. Correctness under non-determinism [1]
   - Forms of non-determinism
   - Non-determinism in concurrent/distributed systems
   - Is consistency/correctness plus non-determinism a contradiction?
   4.2. Select statements [1]
   - Forms of non-deterministic message reception

5. Scheduling [2]
   5.1. Problem definition and design space [1]
   - Scheduling
   5.2. Basic scheduling methods [1]
   - Assumptions for basic scheduling
   - Basic methods

   6.1. Safety properties [3]
   - Schedulability
   - Response time
   - Liveness
   - Safety in concurrent and distributed systems
   6.2. Global state [3]
   - Global state
   - Global state
   - Safety in concurrent and distributed systems
   - Consistency
   - Safety
   - Liveness

7. Architectures for CDS [1]
   7.1. Hardware architecture
   - From microprocessors to microprocessors
   - Pipelining
   - Instructions
   - Systems
   7.2. Software architecture
   - Operating systems
   - Networks
   - Communication
   7.3. Language architecture
   - Expressiveness
   - Syntax
   - Semantics

8. Distributed systems [7]
   8.1. Networks [3]
   - Communication networks
   - Distributed systems
   8.2. Global state [3]
   - Global state
   - Inconsistent processes
   8.3. Distributed algorithms [3]
   - Consistency
   - Distributed algorithms
   - Liveness

9. Distributed safety and liveness [3]
   9.1. Distributed safety [3]
   - Consistency
   - Safety
   - Liveness
   9.2. Distributed liveness [3]
   - Consistency
   - Safety
   - Liveness

    10.1. Transactional memory [2]
    - Correctness
    - Liveness
    - Concurrency
    - Multi-core programs
    - Communication
    - Transactions

Laboratories & Assignments

Laboratories
1. Concurrent programming [3]
   - Acceptance
   - Programs
   2. Concurrent programming
   - Acceptance
   - Programs
   3. Concurrent programming
   - Acceptance
   - Programs

Assignments
1. Concurrent programming [15%]
   - Ada programming task involving:
     - Mutual exclusion
     - Synchronization
     - Message passing
   2. Concurrent programming [15%]
   - Multi-core programming task involving:
     - Inter-process communication
     - Multi-core programming

Examinations
1. Mid-term check [10%]
   - Test question set
   2. Final exam [55%]
   - Exams

Marking
The final mark is based on the assignments (30%) plus the examinations (65%) plus the lab mark (5%).