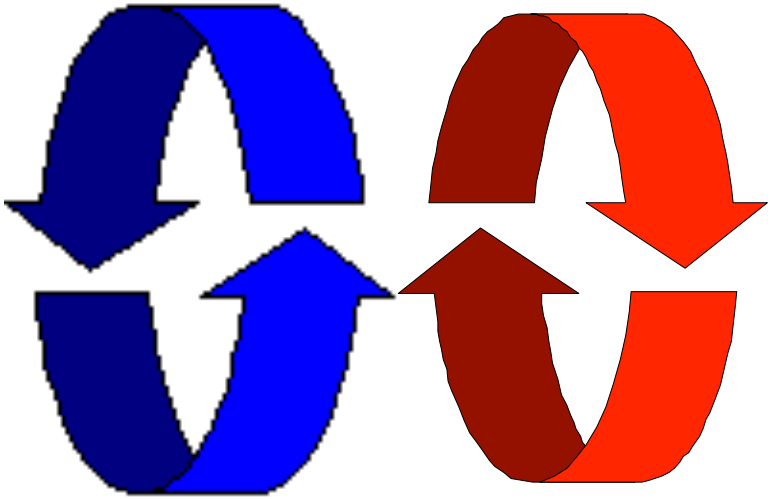
Concurrency

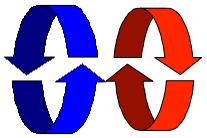
State Models and Programming



Julien Lange Raymond Hu Nobuko Yoshida

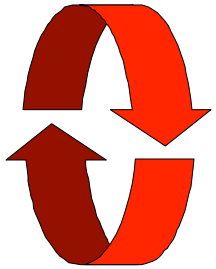
{j.lange, raymond.hu, n.yoshida}@imperial.ac.uk

Logistics

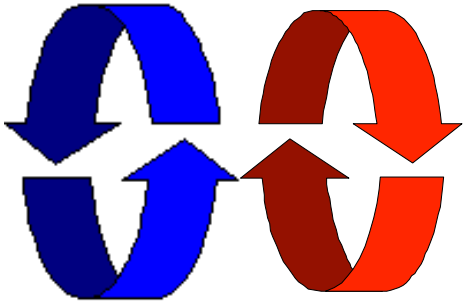


- Lectures:
  - Mondays 11am-12pm, room 311
  - Thursdays 4pm-6pm, room 308
- Labs/Tutorials: Mondays 12pm-1pm (202 + 206)
- TAs: Nick Ng and Alceste Scalas
- Software:
  - LTSA Analyzer (available from the website)
  - Java

What is a Concurrent Program?

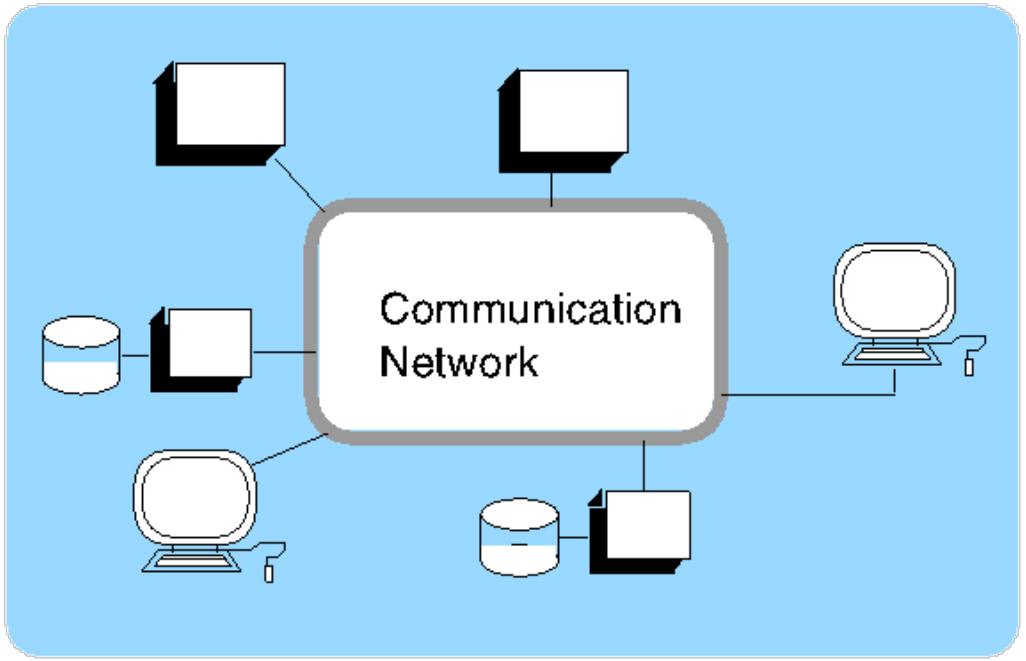


A **sequential** program has a single thread of control.



A **concurrent** program has multiple threads of control allowing it to perform multiple computations in parallel and to control multiple external activities which occur at the same time.

Concurrent and Distributed Software?

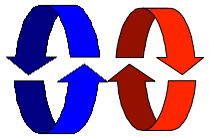


Interacting, concurrent software components of a system:

single machine -> *shared memory interactions*

multiple machines -> *network interactions*

Why Concurrent Programming?



- Performance gain from multiprocessing hardware
  - parallelism.
- Increased application throughput
  - an I/O call need only block one thread.
- Increased application responsiveness
  - high priority thread for user requests.
- More appropriate structure
  - for programs which interact with the environment, control multiple activities and handle multiple events.

Do I need to know about concurrent programming?

Concurrency is widespread but error prone.

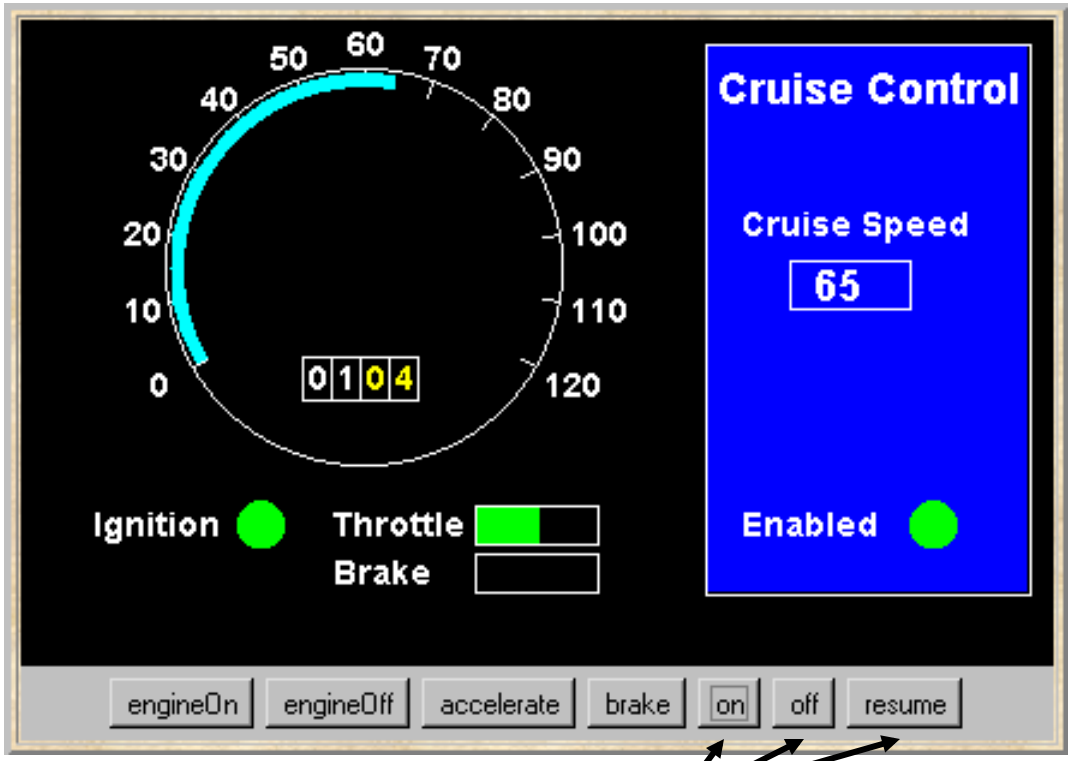
- Therac - 25 computerised radiation therapy machine

  Concurrent programming errors contributed to accidents causing deaths and serious injuries.

- Mars Rover (Pathfinder 1997)

  Problems with interaction between concurrent tasks caused periodic software resets reducing availability for exploration.

Simple example: Cruise Control System



When the car ignition is switched on and the **on** button is pressed, the current speed is recorded and the system is enabled: *it maintains the speed of the car at the recorded setting.*

Pressing the brake, accelerator or **off** button disables the system. Pressing **resume** re-enables the system.

- ***Is the system safe?***
- ***Would testing be sufficient to discover all errors?***

Models

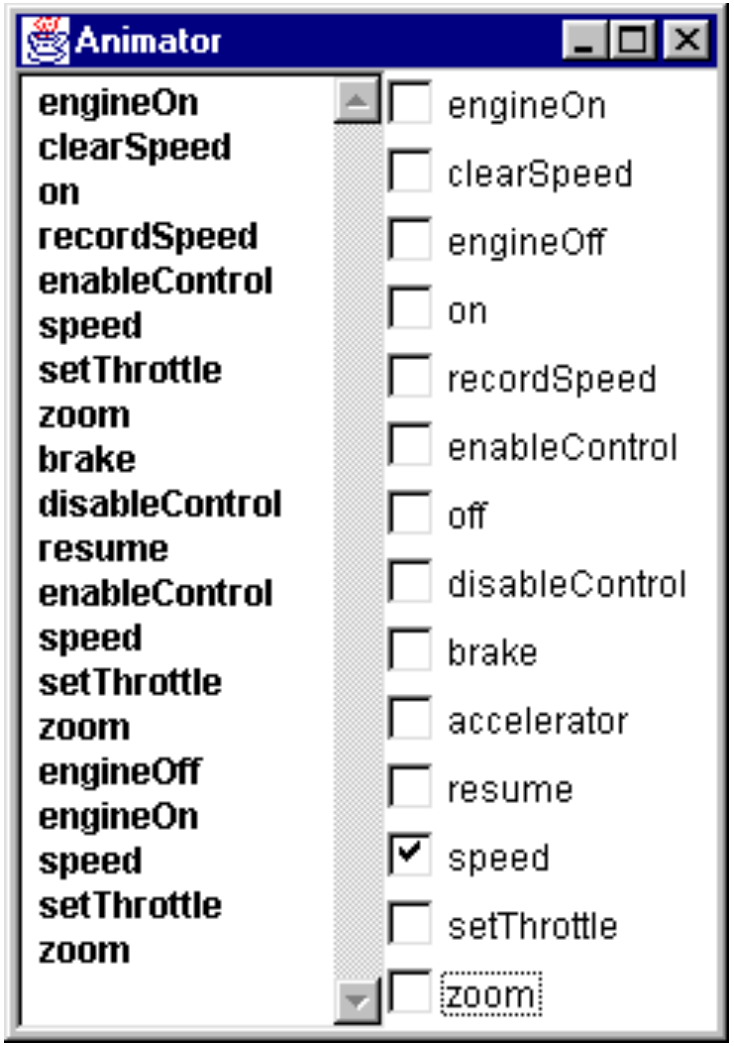
A model is a simplified representation of the real world.

Engineers use models to gain confidence in the adequacy and validity of a proposed design.

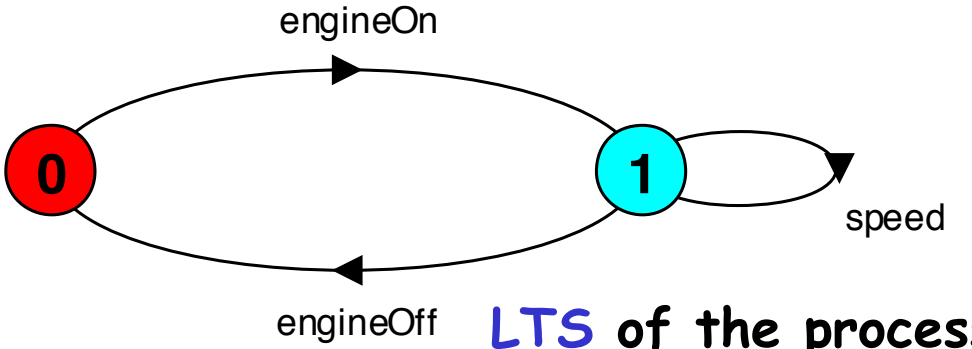
- focus on an aspect of interest - concurrency
- model animation to visualise a behaviour
- mechanical verification of properties (safety & progress)

Models are described using state machines, known as Labelled Transition Systems **LTS**. These are described textually as finite state processes (**FSP**) and displayed and analysed by the ***LTSA*** analysis tool.

Modeling the Cruise Control System



LTSA Animator to step through system actions and events.



LTS of the process that monitors speed.

Later chapters will explain how to construct models such as this so as to perform animation and verification.

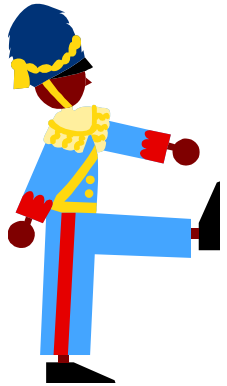
Programming practice in Java

Java is

- widely available, generally accepted and portable
- provides sound set of concurrency features

Hence Java is used for all the illustrative examples, the demonstrations and the exercises. Later chapters will explain how to construct Java programs such as the Cruise Control System.

"Toy" problems are also used as they exemplify particular aspects of concurrent programming problems!



Course objective

This course is intended to provide a sound understanding of the ***concepts***, ***models*** *and* ***practice*** involved in designing concurrent software.

The emphasis on principles and **concepts** provides a thorough understanding of both the problems and the solution techniques. **Modeling** provides insight into concurrent behavior and aids reasoning about particular designs. Concurrent programming in **Java** provides the programming **practice** and experience.

Concrete objectives

- **What this course covers, notably:**
  - Core principles of model-based reasoning for concurrent programs. Building on this you can move on to automata theory, CSP, pi-calculus, model-checking, etc.
  - The synchronization mechanisms underlying any other ("more sophisticated") concurrency library and middlewares.

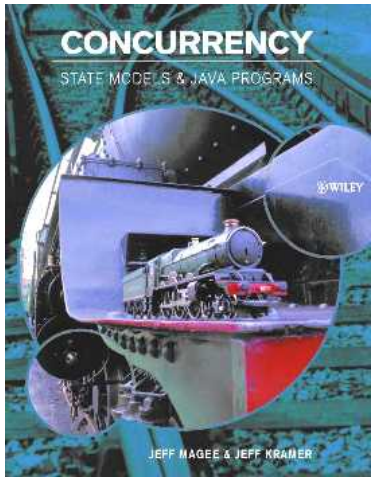
- **Examples of topics we do not cover here:**
  - Event-driven / Message-passing based interactions
  - Concurrent (or lock free) data structures
  - C++ concurrency

Book (http://www.doc.ic.ac.uk/~jnm/book/)

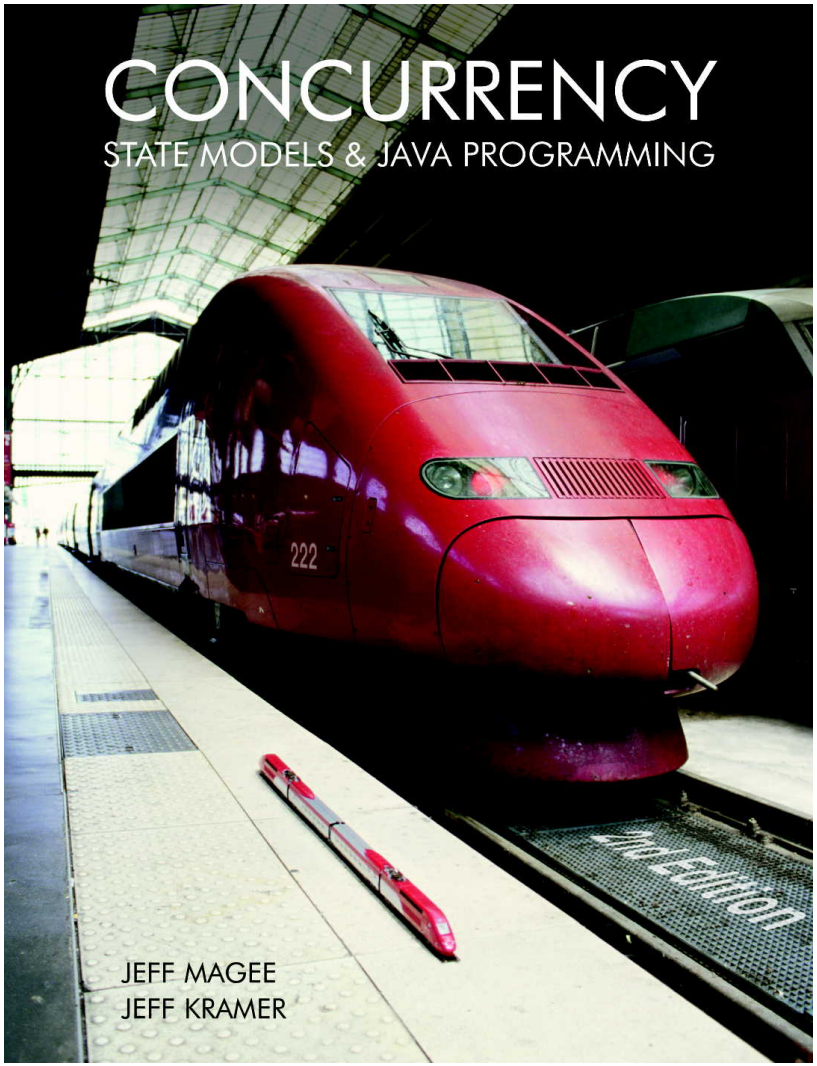
Concurrency: State Models & Java Programs, 2nd Edition

Jeff Magee & Jeff Kramer

WILEY



1st edition



Course Outline

2. Processes and Threads
3. Concurrent Execution
4. Shared Objects & Interference
5. Monitors & Condition Synchronization
6. Deadlock
7. Safety and Liveness Properties

The main basic

Concepts

Models

Practice

Advanced topics …

- Model-based Design
- Dynamic systems
- Message Passing
- Concurrent Software Architectures
- Timed Systems
- Program Verification
- Logical Properties

Web based course material

http://www.wileyeurope.com/college/magee

- Java examples and demonstration programs
- State models for the examples
- Labelled Transition System Analyser (***LTSA***) for modeling concurrency, model animation and model property checking.

Summary

- Concepts
  - we adopt a model-based approach for the design and construction of concurrent programs
- Models
  - we use finite state models to represent concurrent behavior.
- Practice
  - we use Java for constructing concurrent programs.

Examples are used to illustrate the concepts, models and demonstration programs.