Lecture 1 – Conventional SE & Extreme Programming

Karl R. Wilcox
K.R.Wilcox@reading.ac.uk

Course Outline

- **Lectures**
  - 20 Lectures
  - Deepen knowledge in areas already covered
  - Look at new, developing topics
- **Seminars**
  - After lectures (& coffee!)
  - As required
  - Also Career Management Seminars
- **Assessment**
  - By examination (2 hour paper, 70% of coursemark)
  - Group project (3-4 subject TBA, 30% of coursemark)
- **Career Management**
  - 3 assignments
  - Career Profile, Job Study & CV

Objectives

- To remind ourselves of the various process models discussed in part 1
- To contrast these with models with “Extreme Programming”
- To understand the “core practices” of Extreme Programming
- Seminar
  - To discuss the desirability of

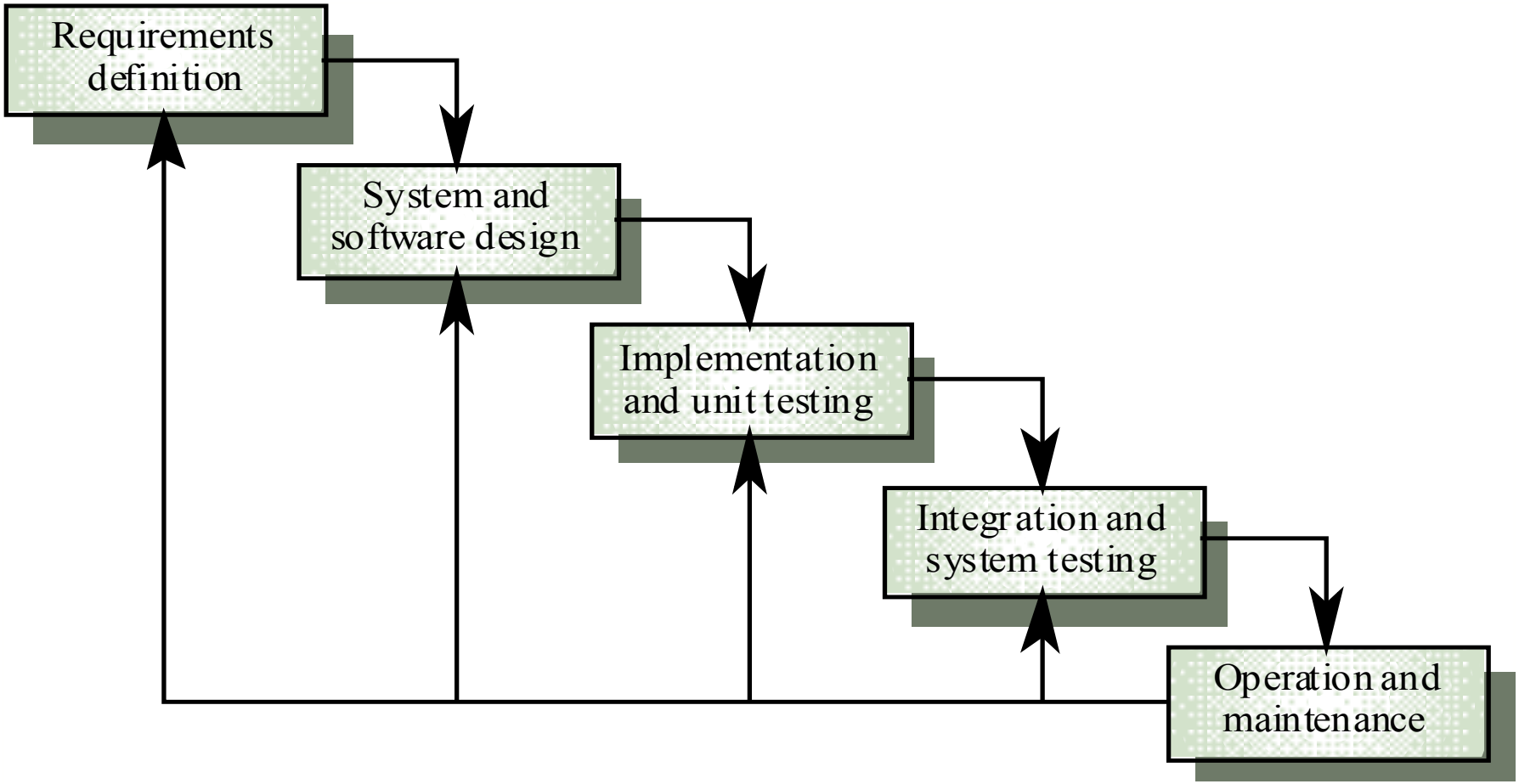
Refresher

- Recall from your first year:
  - Software Development Lifecycles
    - Waterfall, Spiral etc.
  - Activities / stages common to several lifecycles
    - Specification, Design, Implementation etc.
  - Management of Software Development
    - Project Planning, Estimation, Risk etc.
- We might refer to this as “conventional” Software Engineering

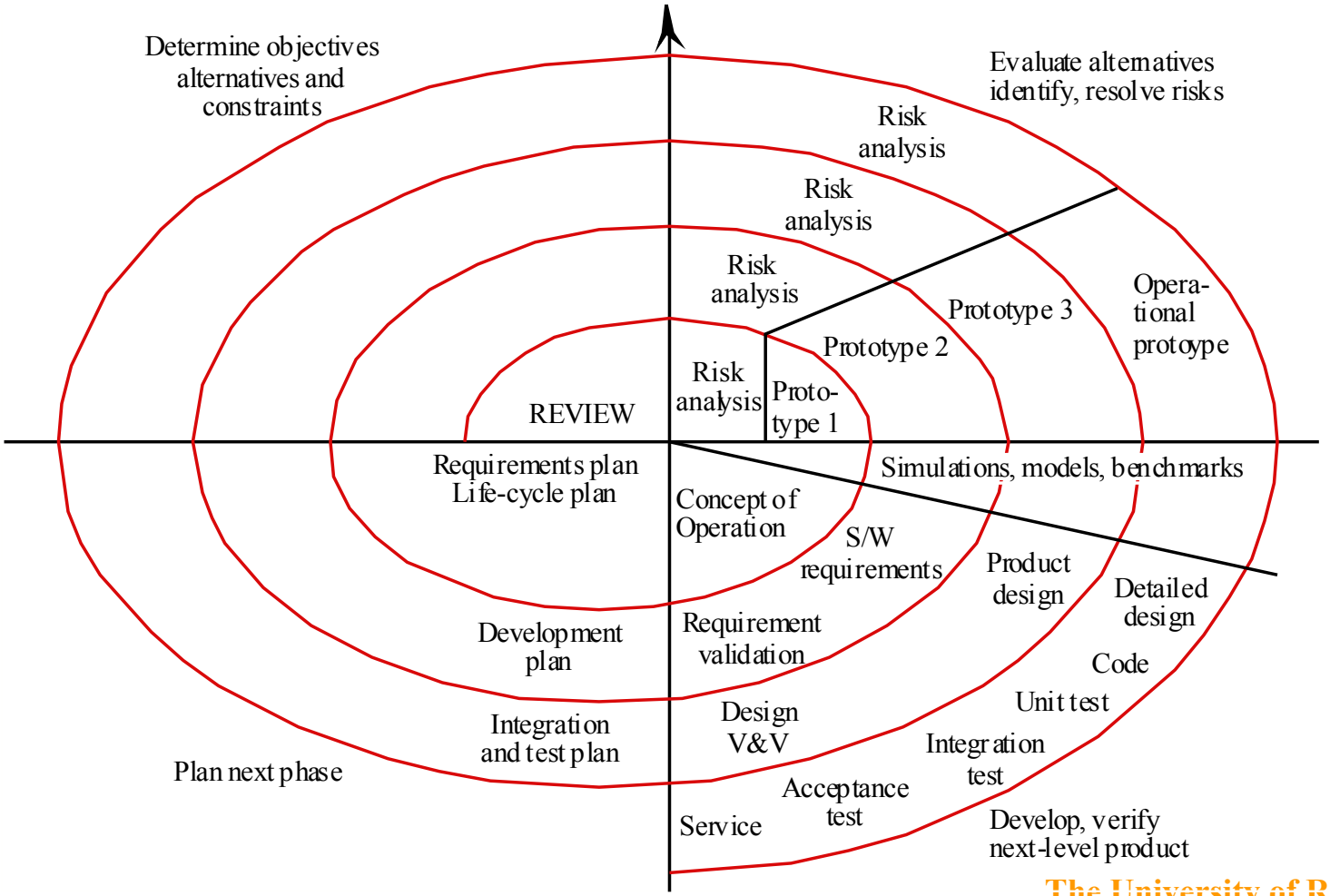
Generic software process models

- **The waterfall model**
  - Separate and distinct phases of specification and development
- **Evolutionary development**
  - Specification and development are interleaved
- **Formal systems development**
  - A mathematical system model is formally transformed to an implementation
- **Reuse-based development**
  - The system is assembled from existing components

Waterfall Development



Spiral Development Model



Discussion of the Spiral Model

- The spiral addresses some of the Waterfall Model problems
  - Allows requirements to change (for later phases)
  - Allows for incremental delivery
  - Explicitly addresses risk

- But there are still issues
  - Requirements (for each phase) need to be stated
  - Phases may still be many months long
  - “Process” driven environment (production line mentality)

Extreme Programming

- **Addresses Process**
  - Acknowledges that it is not possible to capture all requirements, unambiguously, up-front
  - Understands that a process should help a team under pressure, not hinder it
  - Includes lots and lots of feedback and small iterations
- **Addresses People**
  - Acknowledges that a "production line" approach is not always best
  - Gives more trust to developers, "empowerment"
  - Gives more control to developers

Extreme Programming Is NOT

- A "Silver Bullet" for Software Engineering Ills
  - (There aren't any, remember!)
- Suitable for everything
  - Works best in small, co-located teams of 5 – 12 people
    - (maximum of thirty)
  - Not appropriate for life-critical systems
    - Limited traceability / provability
- A substitute for skills
  - Needs good people to operate it
    - Everyone involved, not just developers

12 Core Practices

- XP proposes 12 core practices
- Most, generally accepted as “good practice”
- None of them, on their own, particularly radical
- But they are taken to “extremes”
  - If incremental delivery is good, then
    - New builds to developers every few hours
    - New versions to customers every few weeks
  - If unit testing is good, then
    - Let unit testing drive the coding
- XP often thought of just as “keyboard” sharing
  - But more than programming, “Extreme Development”

Extreme Programming

- **Addresses Process**
  - Acknowledges that it is not possible to capture all requirements, unambiguously, up-front
  - Understands that a process should help a team under pressure, not hinder it
  - Includes lots and lots of feedback and small iterations
- **Addresses People**
  - Acknowledges that a "production line" approach is not always best
  - Gives more trust to developers, "empowerment"
  - Gives more control to developers

12 Core Practices - 1

- **Planning**
  - The business decides dates, scopes, priorities, developers provide size, risk, progress info
- **Small releases**
  - Incremental improvements, in production asap
- **Metaphor**
  - Shared model of system
- **Simple Design**
  - The simplest design that will do the job
- **Frequent Testing**
  - Developers test code, customers test function
- **Refactoring**
  - Limit duplication, improve simplicity & flexibility

12 Core Practices - 2

- **Pair Programming**
  - Continuous peer review
- **Collective Ownership**
  - Developers can change any code at any time
- **Continuous Integration**
  - Frequent system builds (to find problems quickly)
- **Reasonable Scheduling**
  - Relaxed workload
- **On-site customer**
  - Full time member of the team
- **Coding Standards**
  - To encourage communication

Features of XP - 1

- There is NO Requirements Document
  - Cannot be specified completely in advance
  - Describe small units of functionality (few sentences)
    - Known as “stories”
  - Frequent testing, close communication between developer and on-site customer
- Changing Plans
  - Units of implementation and acceptance testing are the “stories”
  - Schedule agreed between developers and customers
  - Measure %-age of stories passing acceptance tests

Features of XP - 2

- Test Driven Development
  - Write a unit test
  - Produce enough code to pass the test
  - Refactor

- Refactoring
  - Make disciplined changes to code to
    - Make it simpler / faster / easier to understand
    - Make it more flexible (for use elsewhere)
  - But make sure it still passes the unit test

Features of XP - 3

- **No architectural model**
  - Total architecture cannot be defined up-front
  - Implement the simplest thing that can possibly work
  - Defer complexity
  - The metaphor (shared model) is the guide to assembly
- **Continuous Integration**
  - V. frequent system builds for developers (hourly)
  - Frequent releases to customers (weekly / monthly)
  - All changes incremental

XP Certification

- Recall from last year, ISO9001 and the SEI Capability Maturity Model
- Some companies have obtained ISO9001 certification with XP teams
- The SEI considers that full XP implementation may equate to CMM Level 3

Summary

- Extreme Programming is said to encourage values of:
  - Courage
  - Feedback
  - Communication
  - Simplicity

- What is your view?

Seminar Question

- What are the implications for each of the following groups? Will they like or dislike the approach?
  - Developers
  - Project Managers
  - Customers
  - Contract Lawyers