Lecture 2 – Software Process Models

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

Objectives

- To introduce software process models
- To describe a number of different process models and when they may be used
- To introduce CASE technology to support software process activities

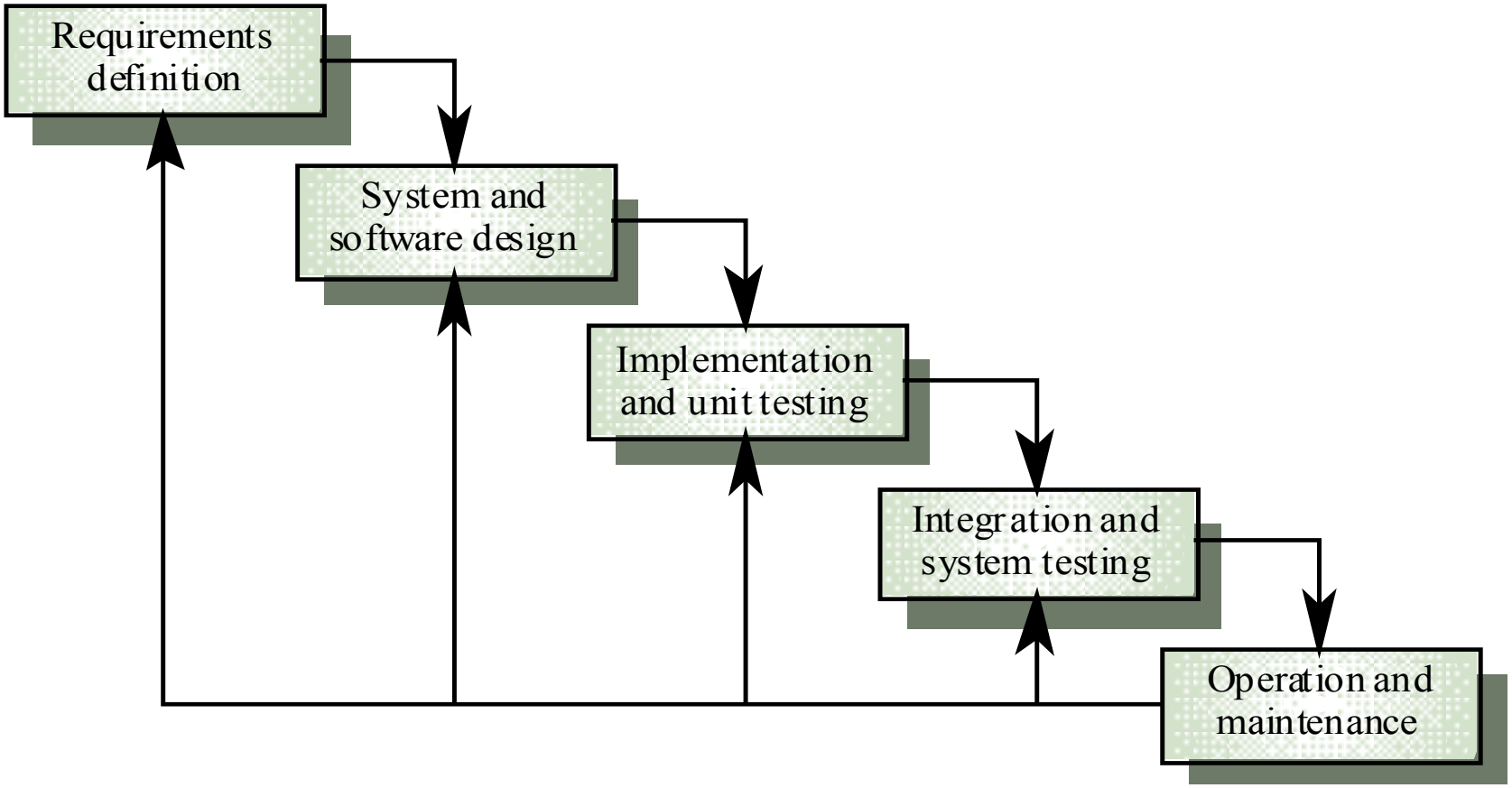
The software process

- A structured set of activities required to develop a software system
  - Specification
  - Design
  - Validation
  - Evolution
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective

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 model



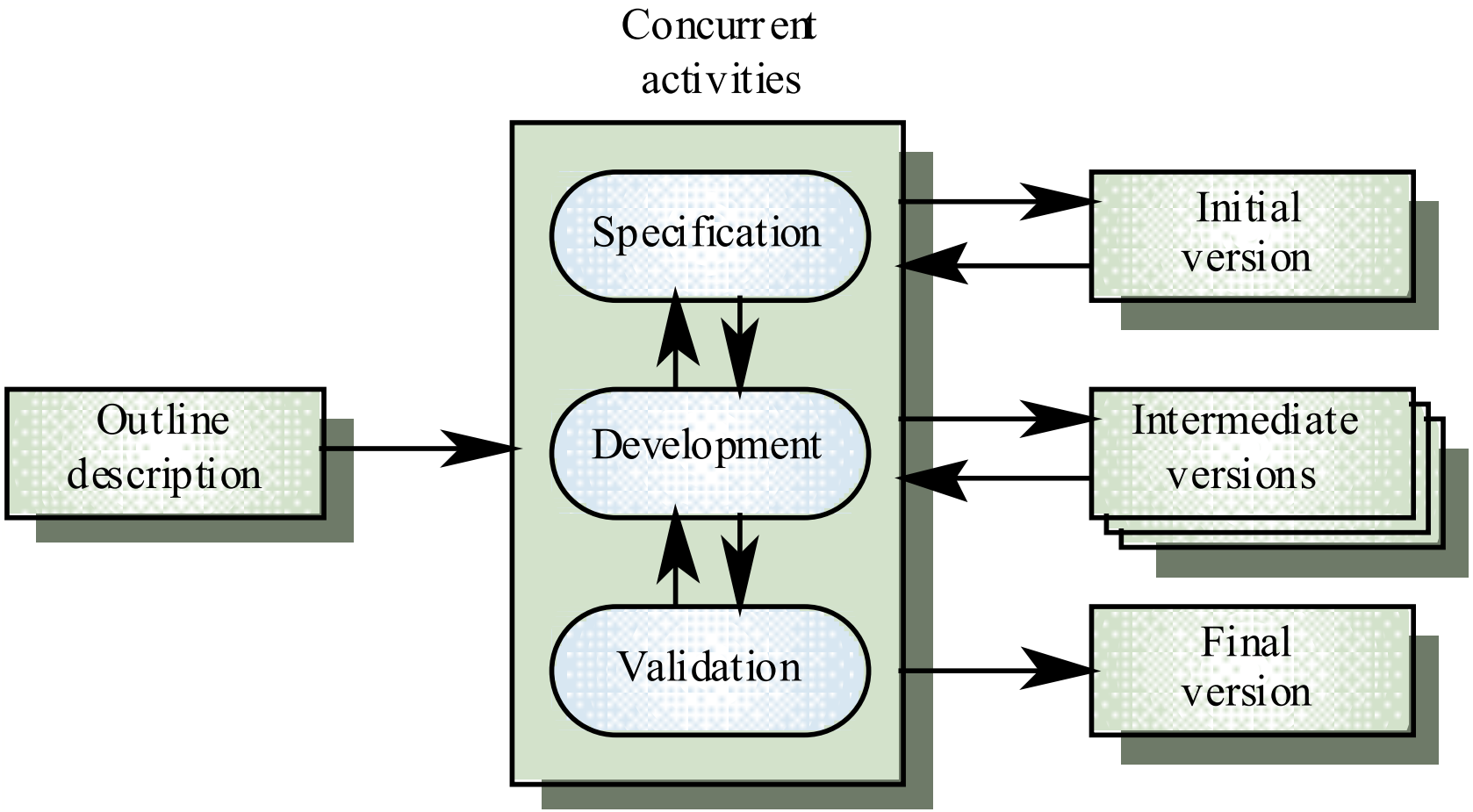
Waterfall model problems

- The drawback of the waterfall model is the difficulty of accommodating change after the process is underway
- Inflexible partitioning of the project into distinct stages
- This makes it difficult to respond to changing customer requirements
- Therefore, this model is only appropriate when the requirements are well-understood

Evolutionary development

- **Exploratory development**
  - **Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements**
- **Throw-away prototyping**
  - **Objective is to understand the system requirements. Should start with poorly understood requirements**

Evolutionary development



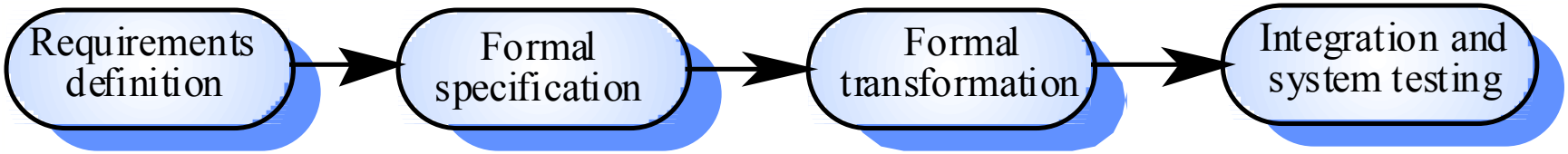
Evolutionary development

- **Problems**
  - Lack of process visibility
  - Systems are often poorly structured
  - Special skills (e.g. in languages for rapid prototyping) may be required
- **Applicability**
  - For small or medium-size interactive systems
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime systems

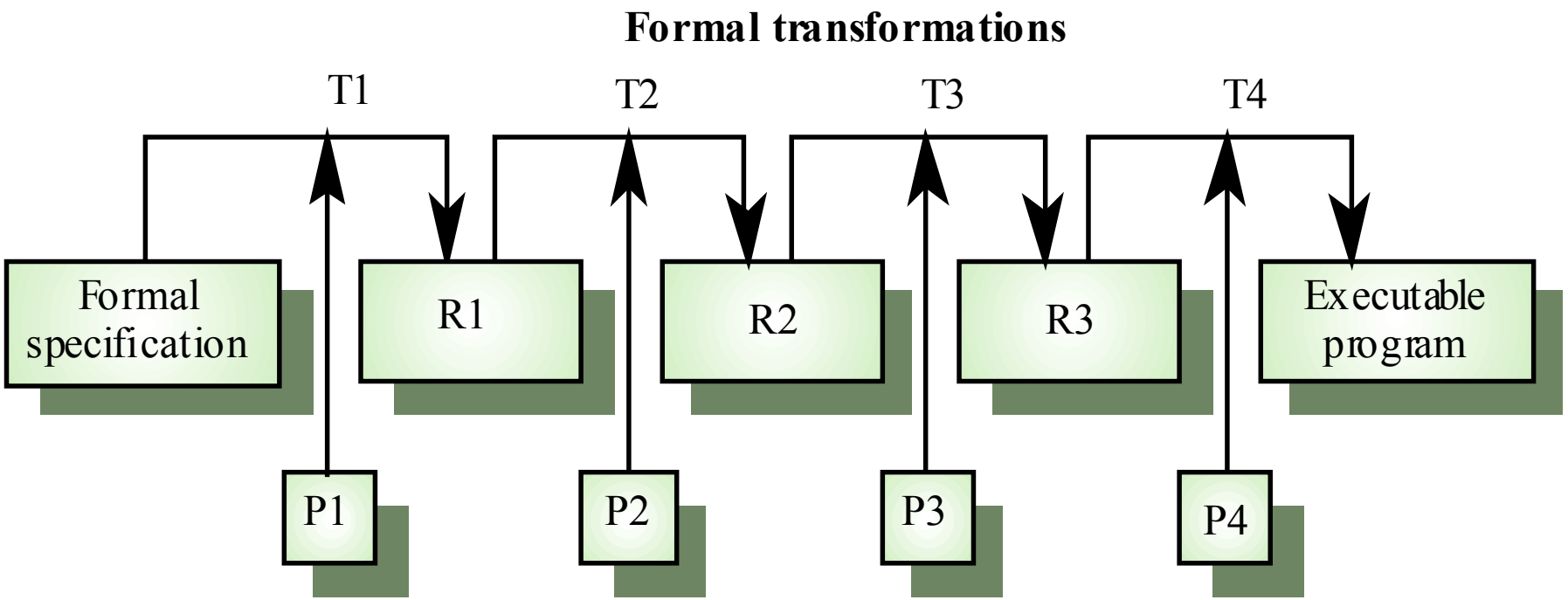
Formal systems development

- Based on the transformation of a mathematical specification through different representations to an executable program
- Transformations are 'correctness-preserving' so it is straightforward to show that the program conforms to its specification
- Embodied in the 'Cleanroom' approach to software development

Formal systems development



Formal transformations



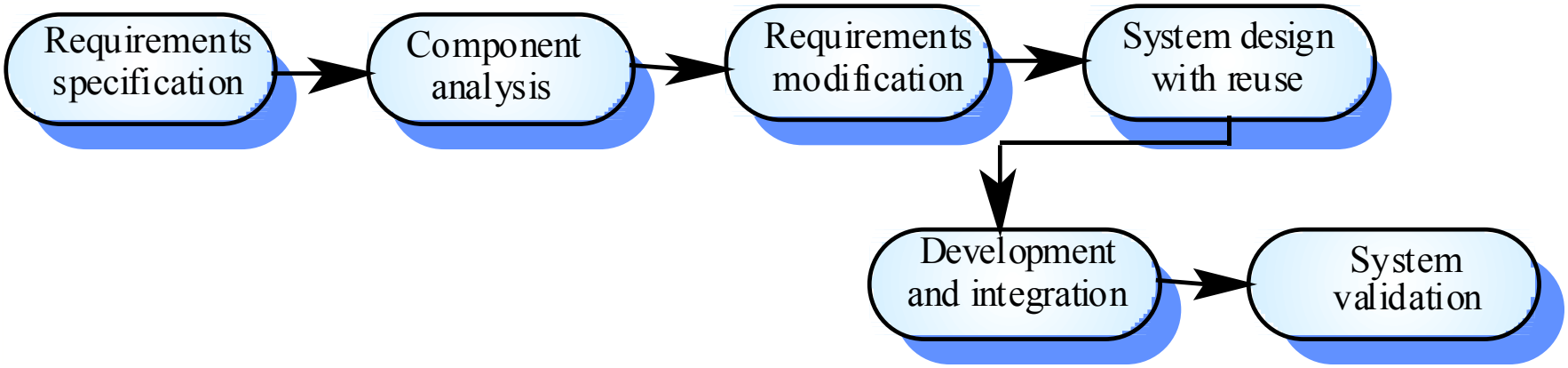
Formal systems development

- **Problems**
  - **Need for specialised skills and training to apply the technique**
  - **Difficult to formally specify some aspects of the system such as the user interface**
- **Applicability**
  - **Critical systems especially those where a safety or security case must be made before the system is put into operation**

Reuse-oriented development

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems
- Process stages
  - Component analysis
  - Requirements modification
  - System design with reuse
  - Development and integration
- This approach is becoming more important but still limited experience with it

Reuse-oriented development



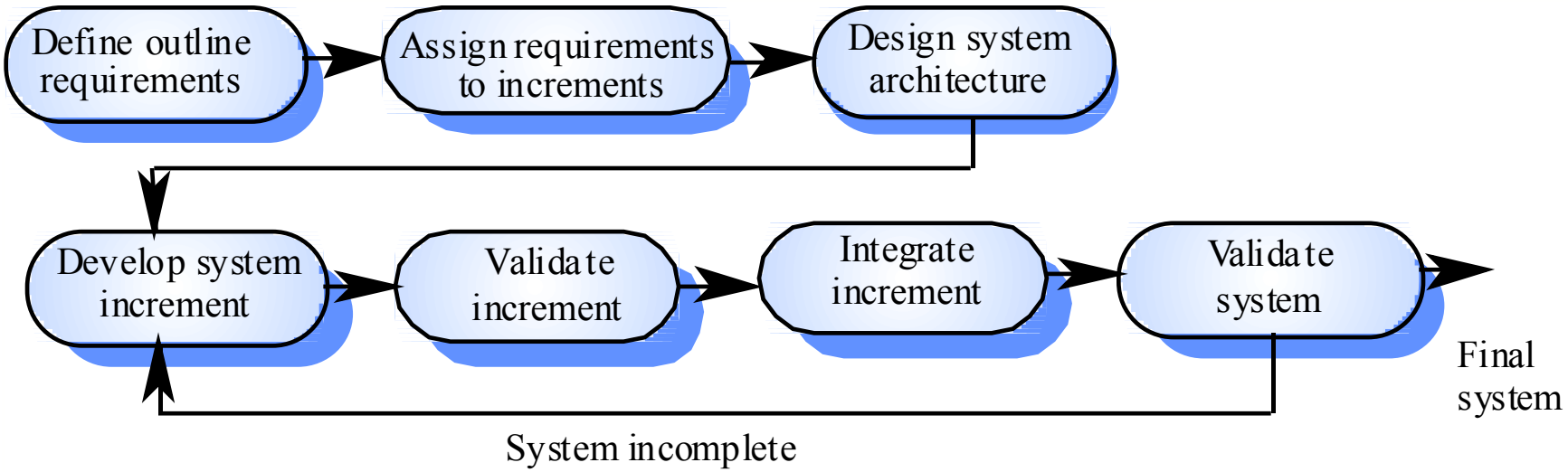
Process iteration

- System requirements ALWAYS evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems
- Iteration can be applied to any of the generic process models
- Two (related) approaches
  - Incremental development
  - Spiral development

Incremental development

- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality
- User requirements are prioritised and the highest priority requirements are included in early increments
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve

Incremental development



Incremental development advantages

- Customer value can be delivered with each increment so system functionality is available earlier
- Early increments act as a prototype to help elicit requirements for later increments
- Lower risk of overall project failure
- The highest priority system services tend to receive the most testing

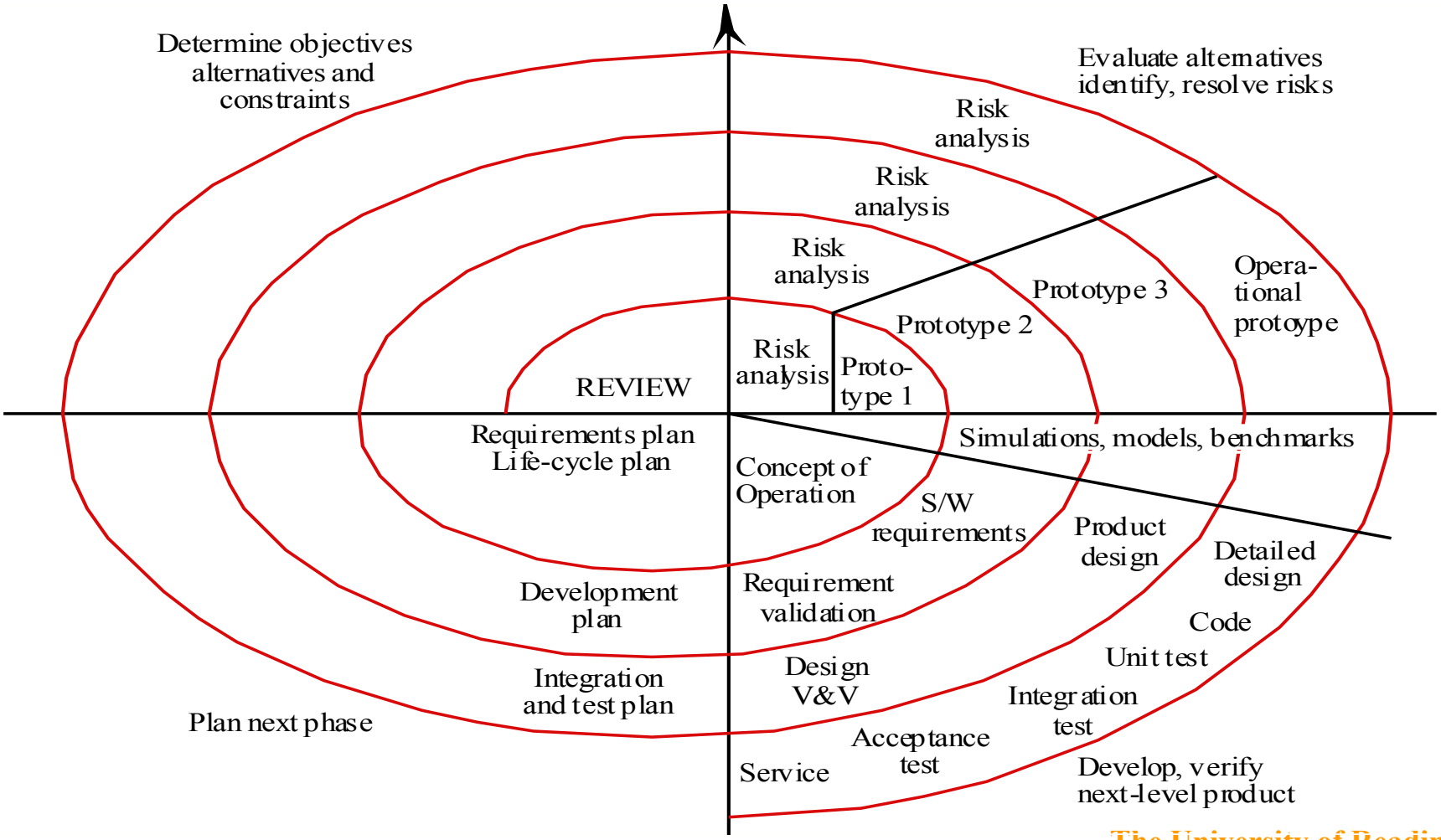
Extreme programming

- New approach to development based on the development and delivery of very small increments of functionality
- Relies on constant code improvement, user involvement in the development team and pairwise programming
- (More next year)

Spiral development

- Process is represented as a spiral rather than as a sequence of activities with backtracking
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
- Risks are explicitly assessed and resolved throughout the process

Spiral model of the software process



Spiral model sectors

- **Objective setting**
  - Specific objectives for the phase are identified
- **Risk assessment and reduction**
  - Risks are assessed and activities put in place to reduce the key risks
- **Development and validation**
  - A development model for the system is chosen which can be any of the generic models
- **Planning**
  - The project is reviewed and the next phase of the spiral is planned

Automated process support (CASE)

- Computer-aided software engineering (CASE) is software to support software development and evolution processes
- Activity automation
  - Graphical editors for system model development
  - Data dictionary to manage design entities
  - Graphical UI builder for user interface construction
  - Debuggers to support program fault finding
  - Automated translators to generate new versions of a program

Case technology

- Case technology has led to significant improvements in the software process though not the order of magnitude improvements that were once predicted
  - Software engineering requires creative thought - this is not readily automatable
  - Software engineering is a team activity and, for large projects, much time is spent in team interactions. CASE technology does not really support these

CASE integration

- **Tools**
  - Support individual process tasks such as design consistency checking, text editing, etc.
- **Workbenches**
  - Support a process phase such as specification or design, Normally include a number of integrated tools
- **Environments**
  - Support all or a substantial part of an entire software process. Normally include several integrated workbenches

Key points

- Software processes are the activities involved in producing and evolving a software system. They are represented in a software process model
- General activities are specification, design and implementation, validation and evolution
- Generic process models describe the organisation of software processes
- Iterative process models describe the software process as a cycle of activities

Key points

- Requirements engineering is the process of developing a software specification
- Design and implementation processes transform the specification to an executable program
- Validation involves checking that the system meets to its specification and user needs
- Evolution is concerned with modifying the system after it is in use
- CASE technology supports software process activities

Seminar Activities

- Comparison between bridge building and IT system building
- Exercise in identifying differences between a program and a product
  - To be handed in (but not assessed) in 3 weeks

Next Two Weeks

- **Tim Millea will cover formal methods**
  - **Recall, formal development, especially formal requirements**

- **Following week we will cover conventional requirements analysis**