Software Processes

- Coherent sets of activities for specifying, designing, implementing and testing software systems

The software process

1. A structured set of activities required to develop a software system
   - Specification
   - Design
   - Validation
   - Evolution
2. 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

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

Waterfall model

Waterfall model phases

1. Requirements analysis and definition
2. System and software design
3. Implementation and unit testing
4. Integration and system testing
5. Operation and maintenance
6. The drawback of the waterfall model is the difficulty of accommodating change after the process is underway
Waterfall model problems

1. Inflexible partitioning of the project into distinct stages
2. This makes it difficult to respond to changing customer requirements
3. Therefore, this model is only appropriate when the requirements are well-understood

Evolutionary development

1. 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
2. Throw-away prototyping
   - Objective is to understand the system requirements. Should start with poorly understood requirements

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

1. Based on the transformation of a mathematical specification through different representations to an executable program
2. Transformations are ‘correctness-preserving’ so it is straightforward to show that the program conforms to its specification

N.B. really about replacing/augmenting/supporting the design and implementation phase of software development
### Formal transformations

- **R1**: Formal specification
- **R2**: Executable program
- **T1**: Proofs of transformation correctness

### Formal systems development

1. **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
2. **Applicability**
   - Critical systems especially those where a safety or security case must be made before the system is put into operation

### Reuse-oriented development

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

### Process iteration

1. 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
2. Iteration can be applied to any of the generic process models
3. Two (related) approaches
   - Incremental development
   - Spiral development

### Incremental development

1. 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
2. User requirements are prioritised and the highest priority requirements are included in early increments
3. Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve
Incremental development

- Define outline requirements
- Validate requirements
- Develop system requirements
- Validate system requirements
- Integrate system
- Validate system
- Final system

System incomplete

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

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

- 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
summary
normative process models

- waterfall model
- evolutionary development
- formal development
- reuse-oriented development

mainly effect design and implementation

summary
similar activities

- requirements and specification
  - design and implementation
    - architectural design, detailed and sub-system design, integration of components, deployment
  - testing, verification and validation
  - evolution
    - deployment, maintenance, changing requirements

summary
... but different timings

- waterfall model
  - each activity in sequence
  - whole system within each activity
- incremental development
  - each 'slice' of system in sequence
  - all activities for each part
- spiral development
  - when it seems right!

common theme

documents and activities
  (software quality)

stages and phases
  (management)