Software Processes

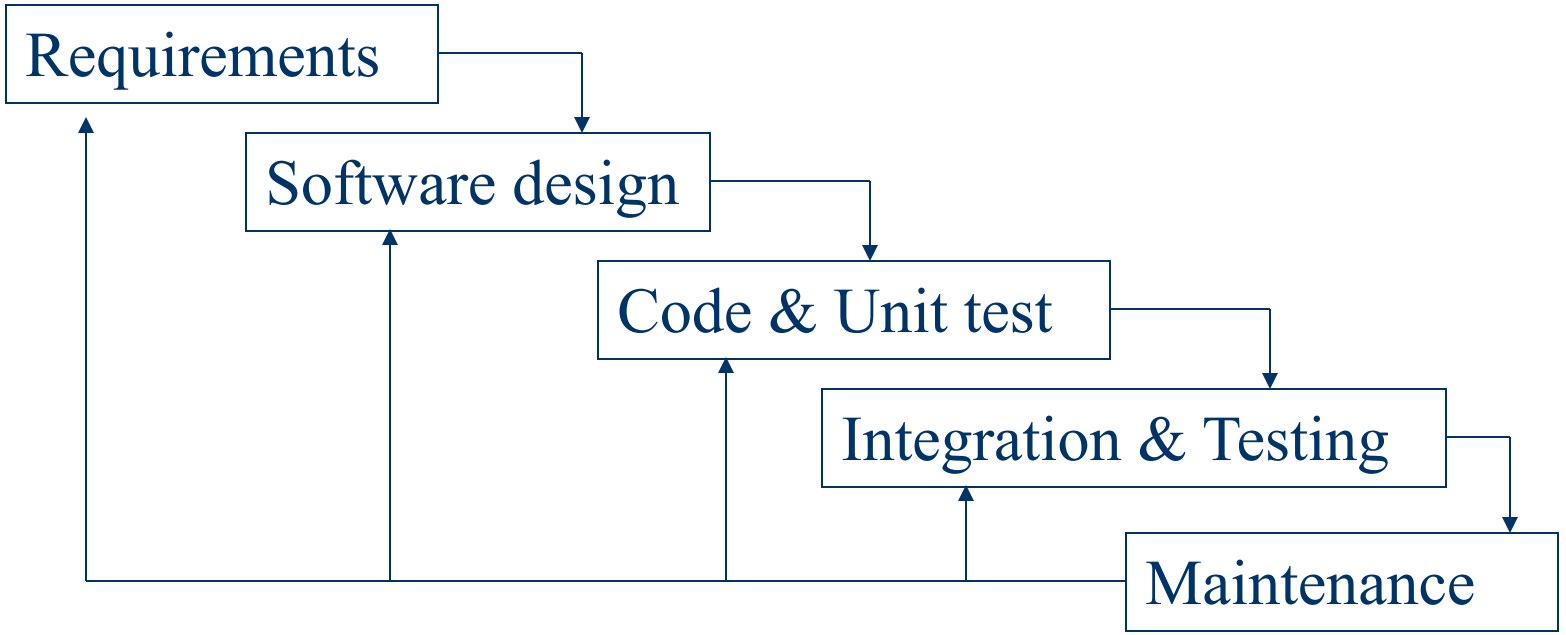
- A Software process is a set of activities and associated results which lead to the production of a software product.
- Activities Common to all software processes:
  - Software specification (initiation, requirements),
  - Software design and implementation,
  - Software validation (testing, QA),
  - Software evolution (enhancement/maintenance)

Software Process Models

- A software process model is an abstract representation of a software (i.e. a roadmap)
- Software process models:
  - Waterfall model
  - Evolutionary development
  - Formal systems development
  - Reuse-based development
- Hybrid software process models:
  - Incremental development
  - Spiral development

Waterfall model

- Also referred to as “Life cycle”, conducted in five stand-alone phases:



Waterfall model: Plus and Minuses

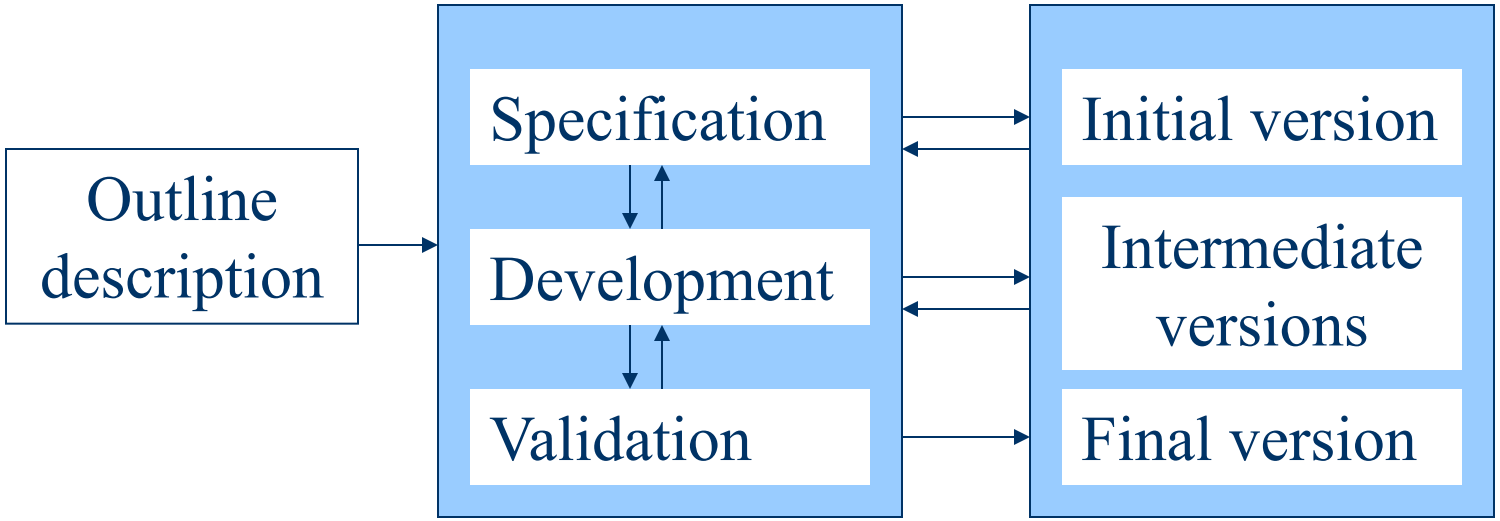
- Advantages of the model:
  - Simple to follow
  - Relatively simple to track progress
  - Good structural design
- Challenges:
  - In practice, often phases overlap
  - Hard to modify and implement changes
  - Need complete requirements from customers to start (the biggest challenge)

Waterfall model Examples

- **enterprise applications**
  - Human Resource Management Systems
  - Supply Chain Management Systems
  - Customer Relationship Management Systems
  - Inventory Management Systems

Evolutionary development

- Develop an initial implementation, expose to users comments, refine until satisfied:



Evolutionary development types

There are two types of evolutionary development:

- Exploratory development
  - Start with requirements that are well defined
  - Add new features when customers propose new requirements
- Throw-away prototyping
  - Objective is to understand customer's requirements (i.e. they often don't know what they want, hence poor requirements to start
  - Use means such as prototyping to focus on poorly understood requirements, redefine requirements as you progress

Evolutionary development: advantages and challenges

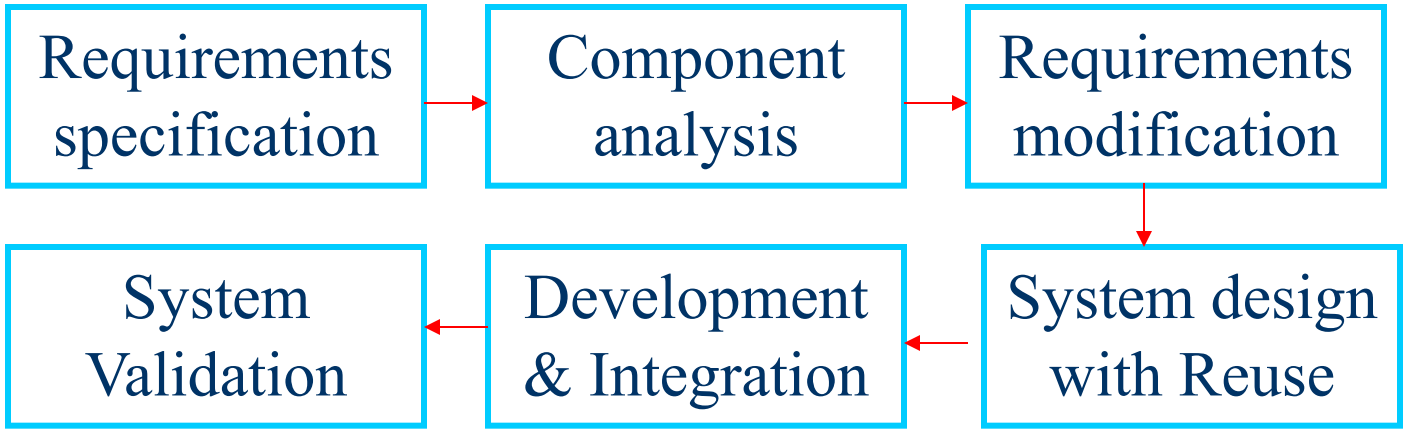
- Advantages:
  - Happier customers since you help them define requirements
  - Flexibility in modifying requirements
  - Prototypes are very visual, hence no ambiguities
- Challenges:
  - Hard to trace the “process” due to the ad-hoc nature
  - Systems are often poorly structured
  - Special tools and techniques may be required (for rapid development) that may be incompatible
  - Not cost-effective to produce documents

Formal systems development

- Development is based on formal mathematical transformation of system specs.
- Software requirements specifications is expressed in a mathematical notion.
- Design, implementation and testing are replaced by a series of mathematical transformations.
- Usually used where safety is highly required.
- Not often used

Reuse–oriented development

- Reuse-oriented approach relies on a large base of reusable software components!
- Design system to capitalize on the existing components



Reuse–oriented : Plus and Minuses

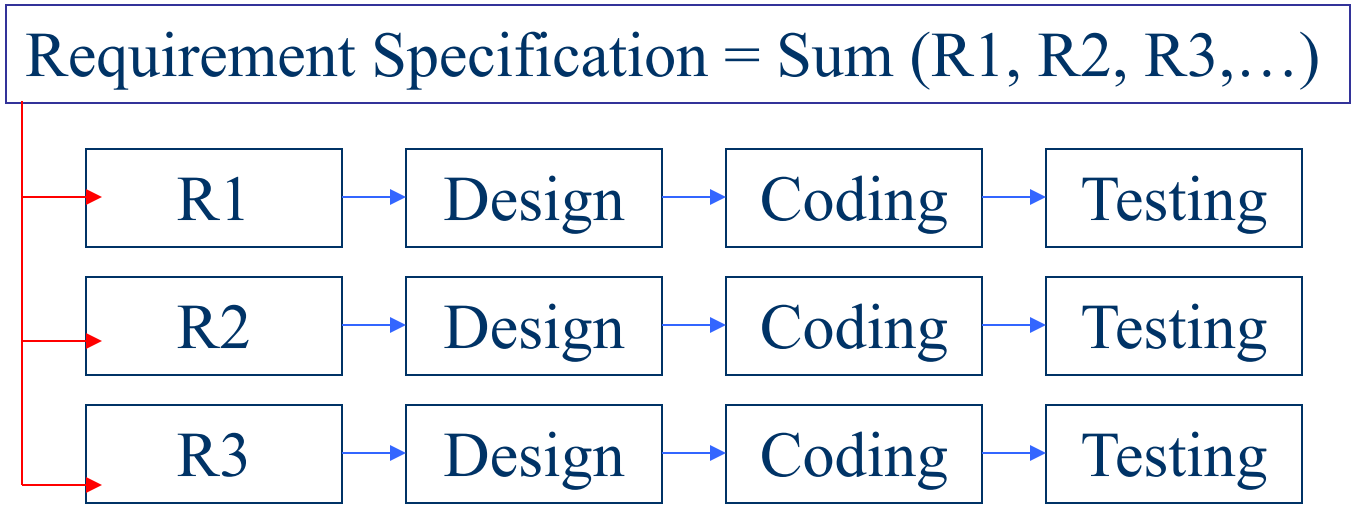
- Advantages:
  - Reduced cost and risk
  - Fast delivery
- Challenges:
  - Requires a large enough component base
  - Some control over the system evolution is lost as new versions of reusable components are not under the control of organization using the component
  - Potential issues in backward/forward compatibility

Reuse–oriented

- **Application system reuse** – The whole application system may reuse without changing into other systems. kinds of requirements.
- **Component reuse** – This may reuse Components of an application.
- **Objects and functions reuse** – Software components that implement a single function, or an object class, can reuse.

Incremental development

- A hybrid model where the software specification, design, implementation, and testing is broken down into a series of increments which are developed and delivered



Incremental development: Advantages and Challenges

- Advantages:
  - Products delivered incrementally hence faster
  - Lower risk of overall project failure
  - Requirements are implemented based on priority
- Challenges:
  - Relationship between different increments may be cumbersome or non-cohesive
  - Size of each increment and the number of increments may cause challenges

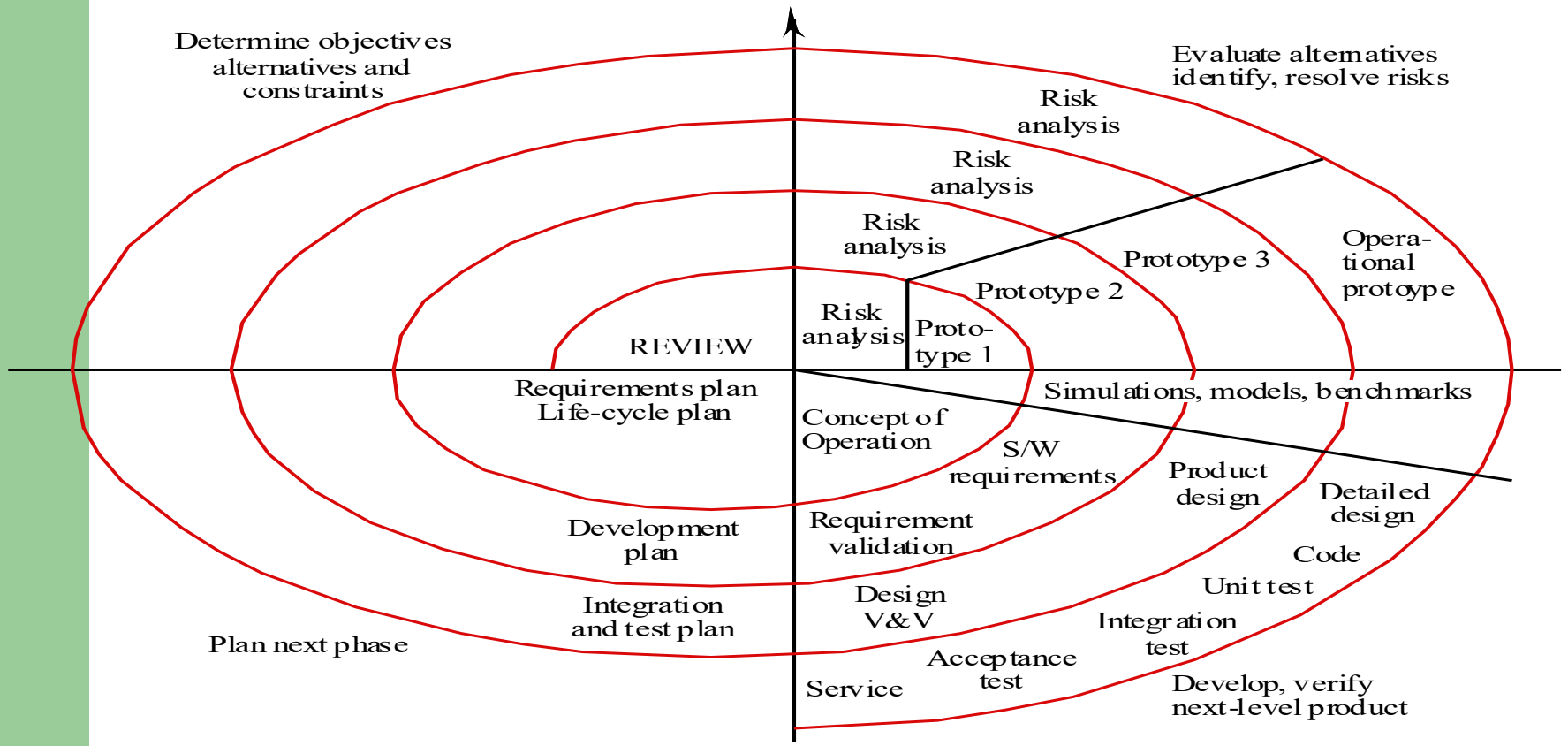
Spiral development

- A hybrid model where the development of the system spirals outward from an initial outline through to the final developed system
- Each loop in the spiral represents a phase of the software process
- Example: the innermost loop might be concerned with system feasibility, next loop with system requirements, next loop with system design and so on.

Spiral development loops

- Each loop in the spiral is split into four sectors:
  1. Object Setting: set specific object for that phase
  2. Risk assessment and reduction
  3. Development and validation: select a development model based on risk levels
  4. Planning: decide if a next loop is required

Spiral Development



Spiral development: Plus & Minuses

- Advantages:
  - Explicit consideration of risks (alternative solutions are evaluated in each cycle)
  - More detailed processes for each development phase
- Disadvantages:
  - Cost
  - Sometime difficult to implement or too time consuming

Software specification activities

1. Feasibility study: doable? Cost-effective? Market timing appropriate?
2. Requirements analysis: derive requirements based on existing system, customer input, marketing information, and all other sources
3. Requirements specification: create formal requirements document
4. Requirement validation: check requirements for realism, consistency and completeness

Software design & implementation activities

1. Architectural design (the big picture)
2. Abstract specification (for each sub-system)
3. Interface design (with other sub-systems)
4. Component design
5. Data structure design
6. Algorithm design
7. Implementation environment selection

Software validation activities

1. Unit testing
2. Module testing
3. Sub-system testing
4. System testing
5. Acceptance testing

Computer-Aided Software Engineering (CASE)

- CASE is the name given to software that is used to support software process activities such as requirements engineering, design, program development and testing.
- CASE tools include:
  - Design editors, compilers, data dictionaries, debuggers, and other system building tools…