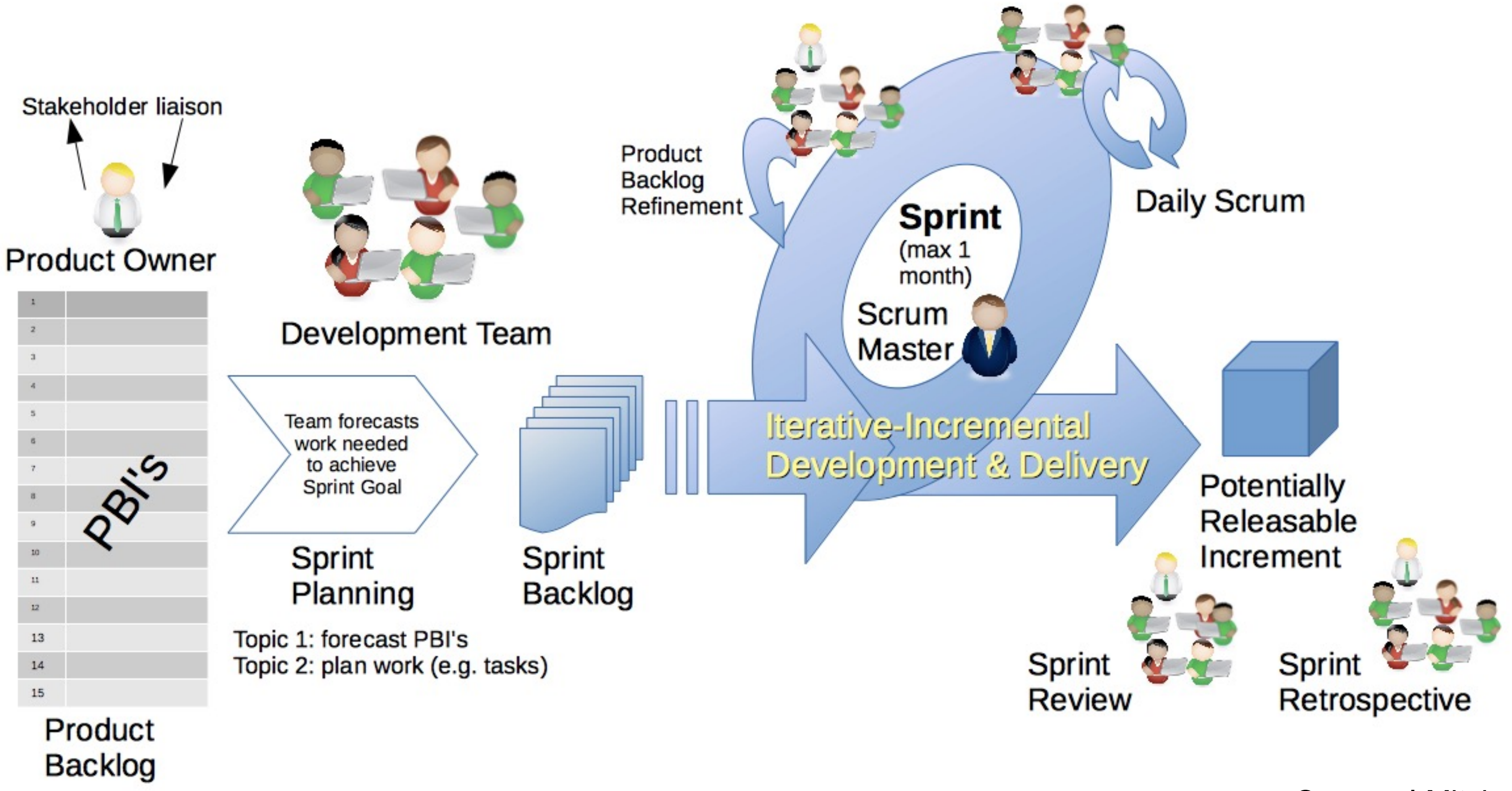
Scrum (Agile)

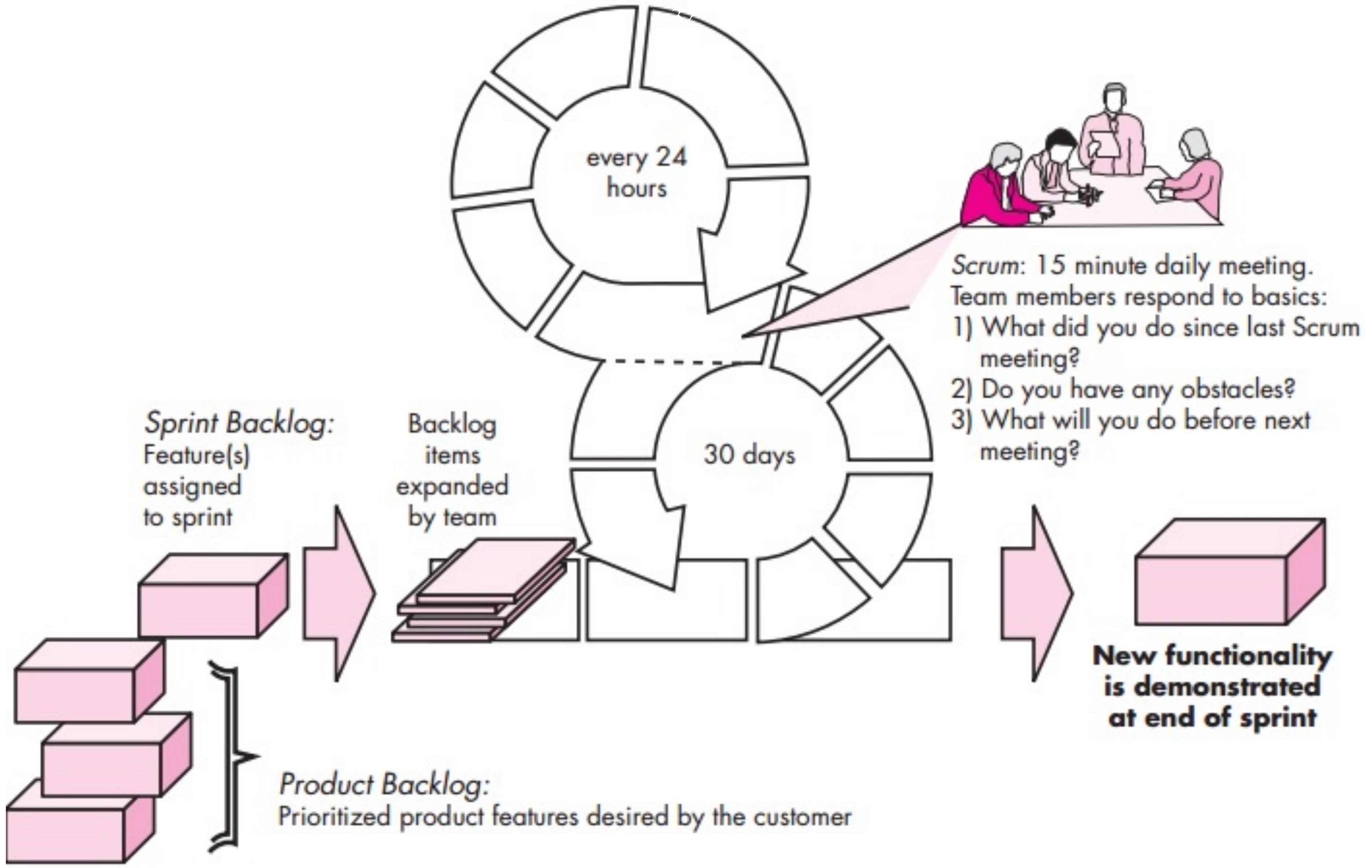
- -Development work is partitioned into “**packets**”
- **-Testing and documentation are on-going** as the product is constructed
- -Increments are made into “**sprints**” and is derived from a “**backlog**” of prioritised requirements
- **-(daily 15-min) meetings,** often casual- may get conducted without chairs
- -“**Demos**” are delivered to the customer within the allocated time-frame

Scrum Framework



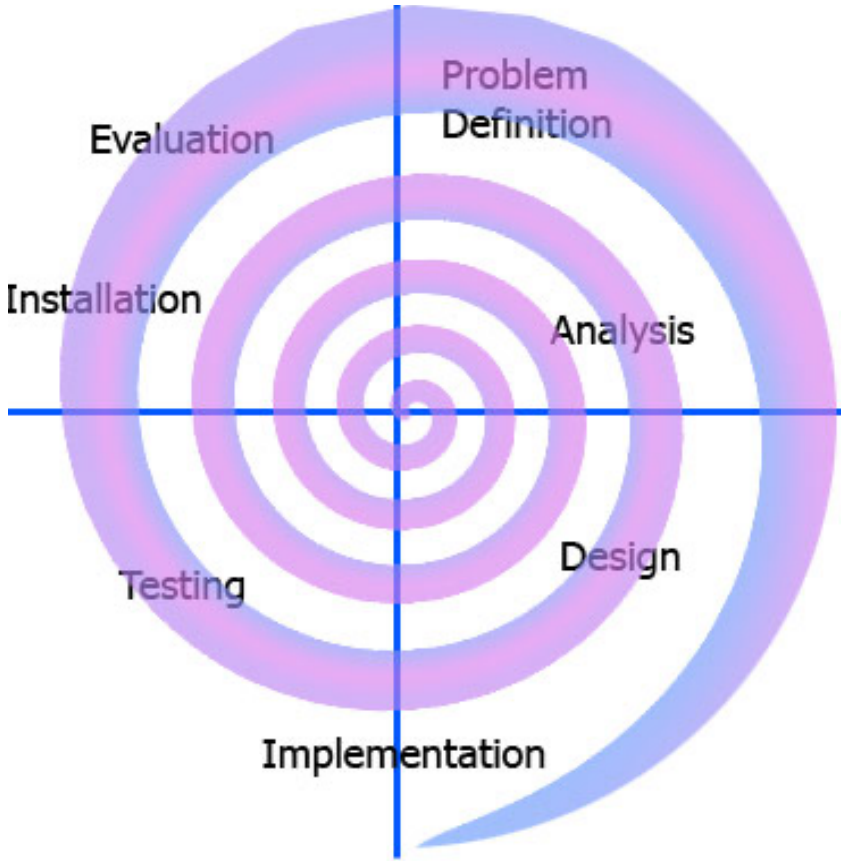
Source: I Mitchell

Scrum Process Flow

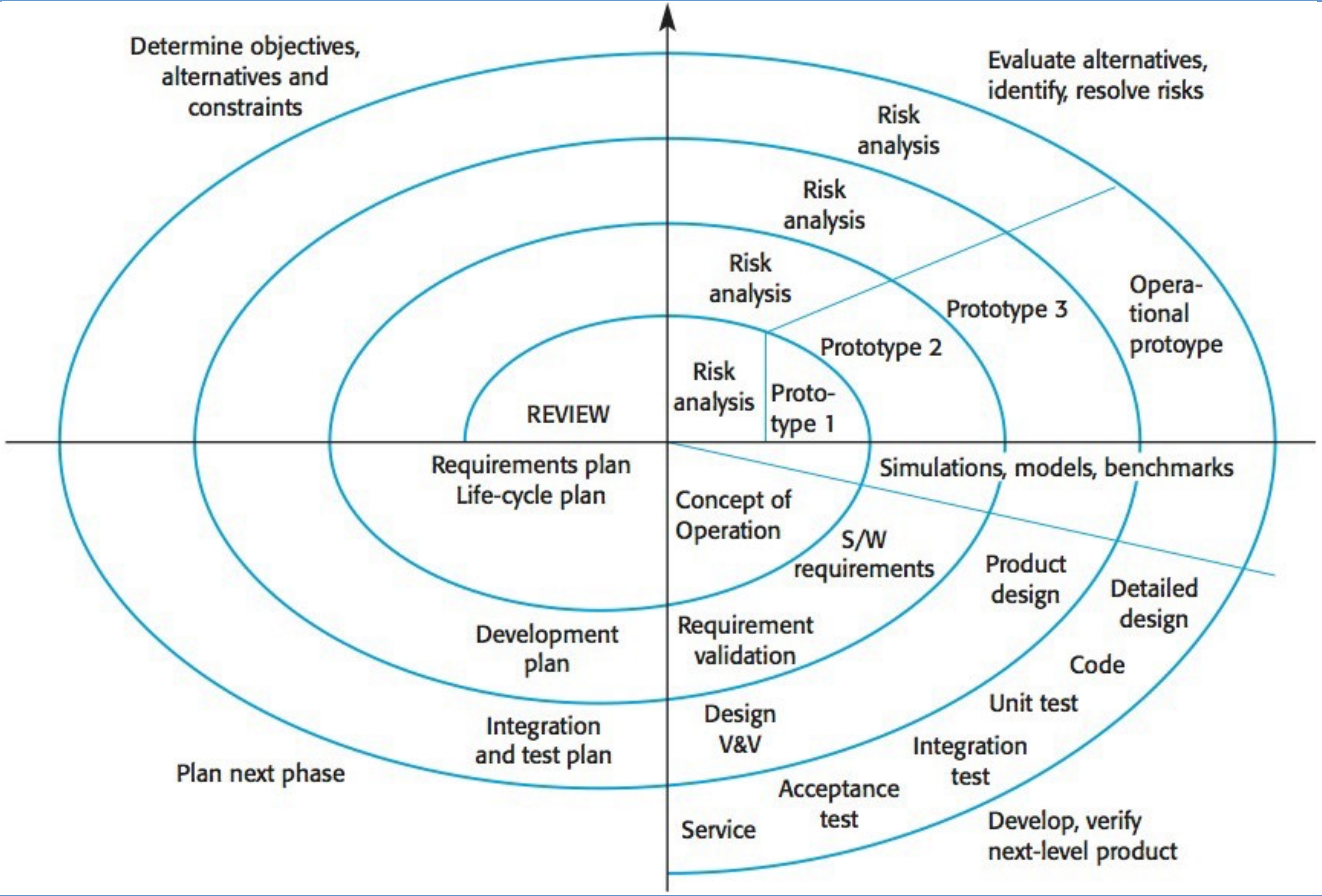

every 24 hours
30 days
Scrum: 15 minute daily meeting.
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog: Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog: Prioritized product features desired by the customer

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

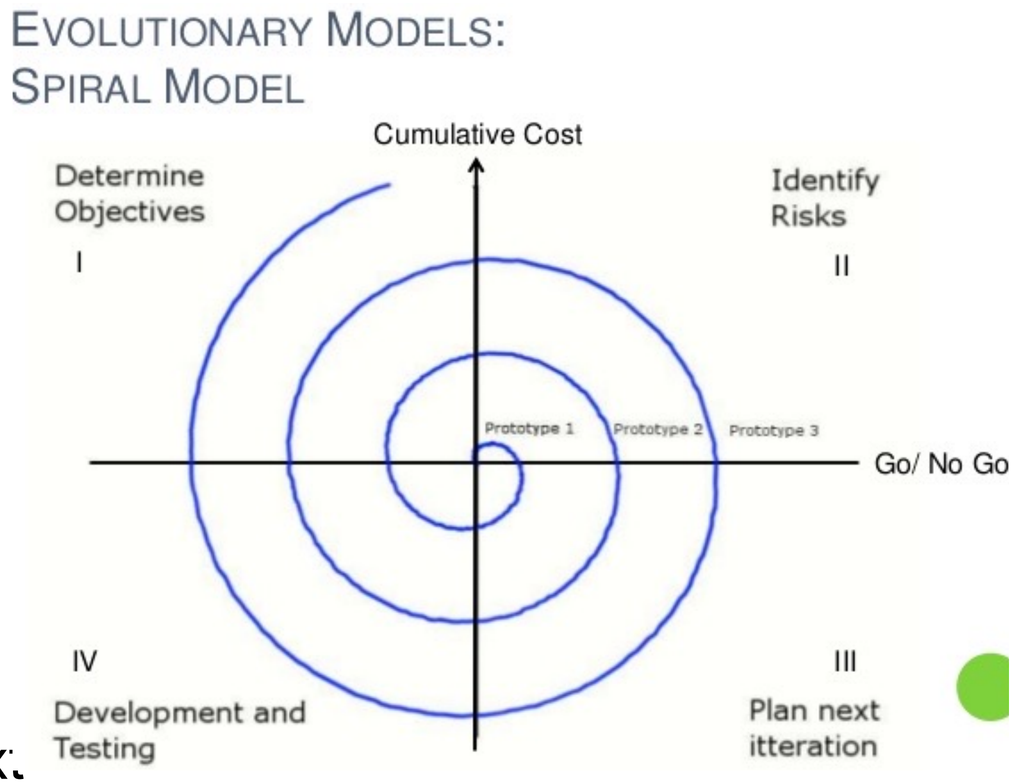
Risks are assessed and activities put in place to reduce the key risks

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 nex: phase of the spiral is planned



Software Process

Fundamental/Core Activities

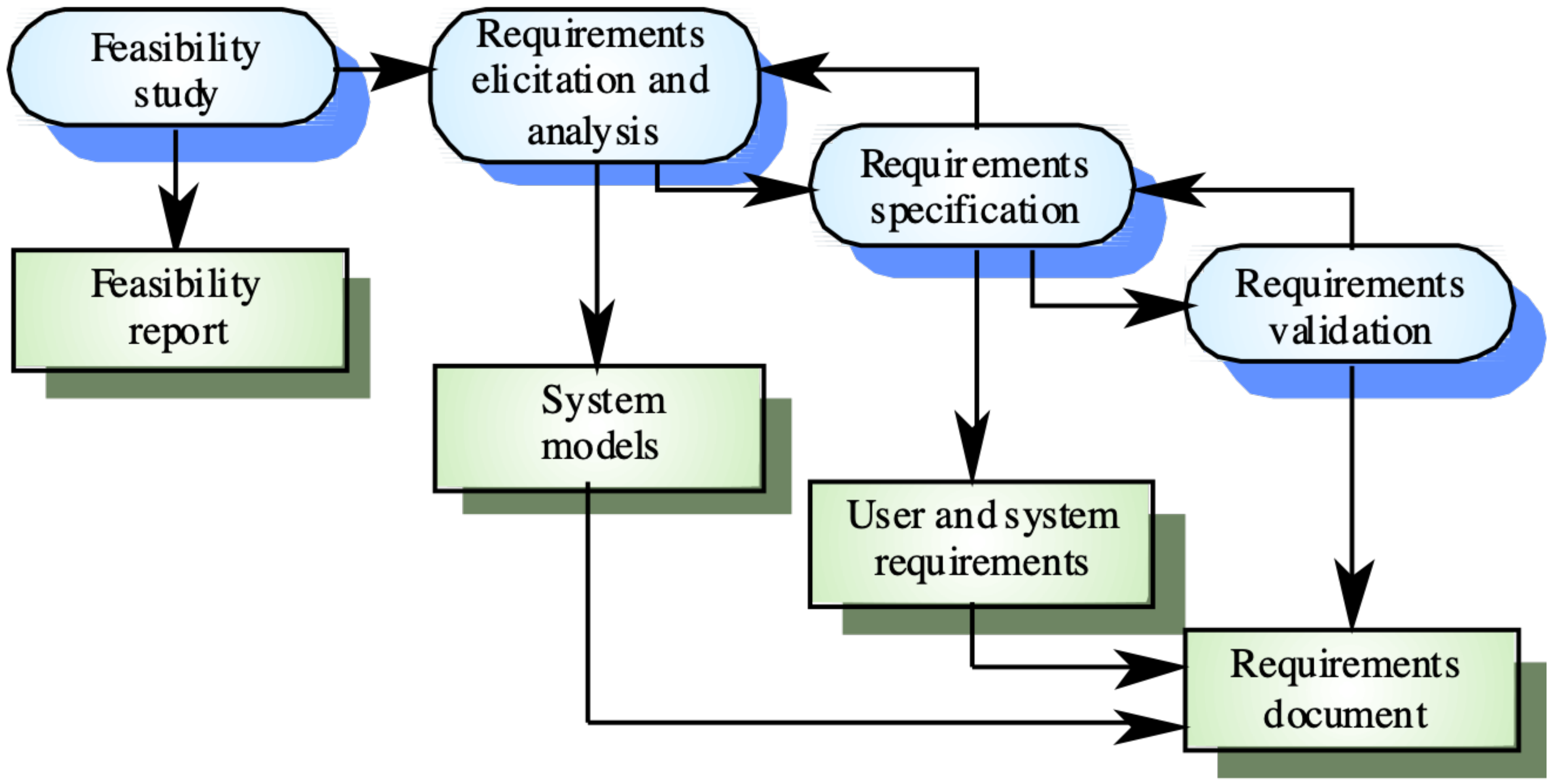
I. Software specification

The process of establishing what functions are required and the constraints on the system's operation and development

Requirements engineering process

- Feasibility study
- Requirements elicitation and analysis
- Requirements specification
- Requirements validation

The requirements engineering process



II. Software design and implementation

The process of converting the system specification into an executable system

Software design

Design a software structure that realises the specification

Implementation

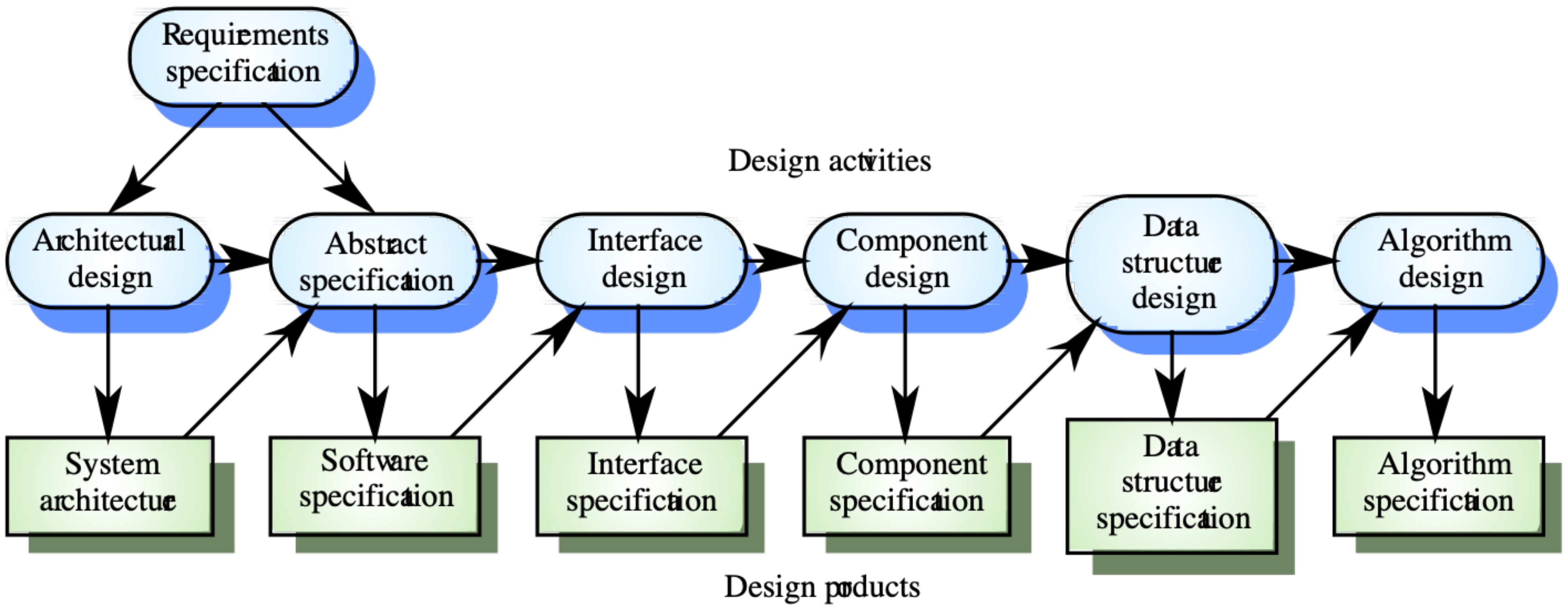
Translate this structure into an executable program

The activities of **design** and **implementation** are closely related and **may be inter-leaved**

Design process activities

- Architectural design
- Abstract specification
- Interface design
- Component design
- Data structure design
- Algorithm design

Design Process



Design methods

Design methods are systematic approaches to developing a software design

The design is usually represented as a set of graphical models

Possible models

- Data-flow model
- Entity-relation-attribute model
- Structural model
- Object models

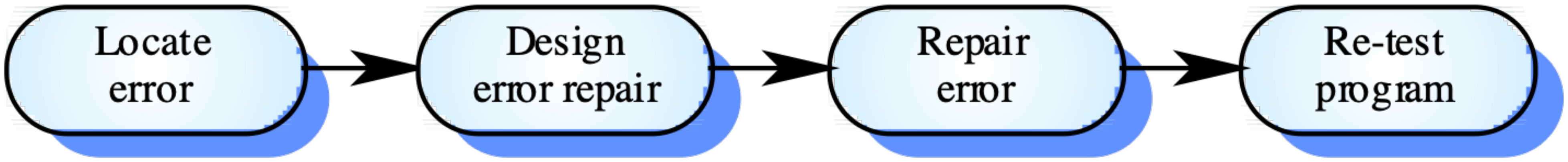
Implementation: Programming and debugging

Translating a design into an executable program and removing errors from that program

Programming is a personal skill-based activity - there is no generic programming process

Programmers carry out some program testing to discover faults in the program and remove these faults in the debugging process

The debugging process



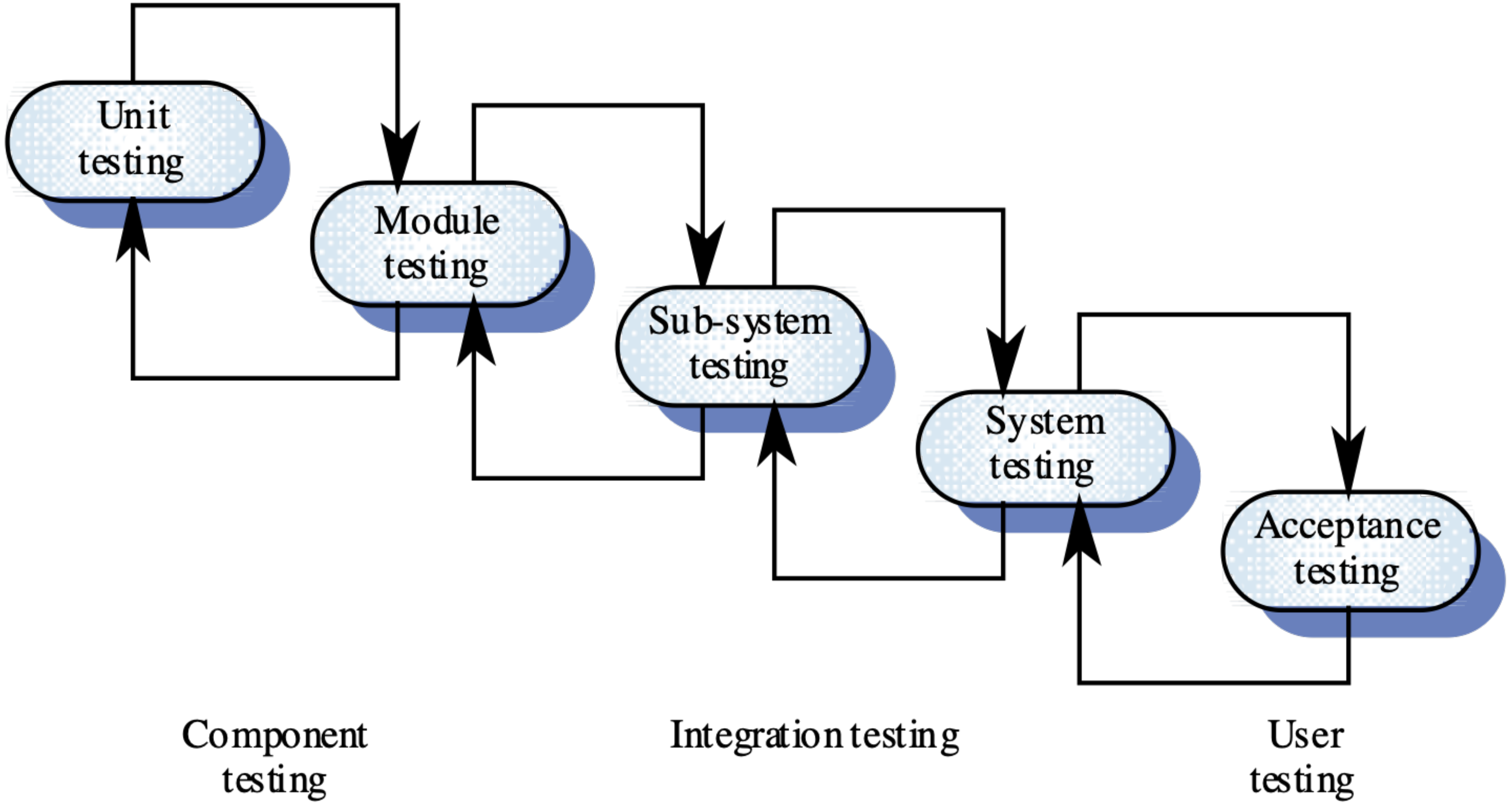
III Software validation

- **Validation, includes verification and evaluation,** is intended to show that a system conforms to its specifications and meets the requirements of the system's customer
- Involves **checking** and **review-processes** and **system testing**
- **System testing involves executing the system with test cases** that are derived from the specification of the real data to be processed by the system

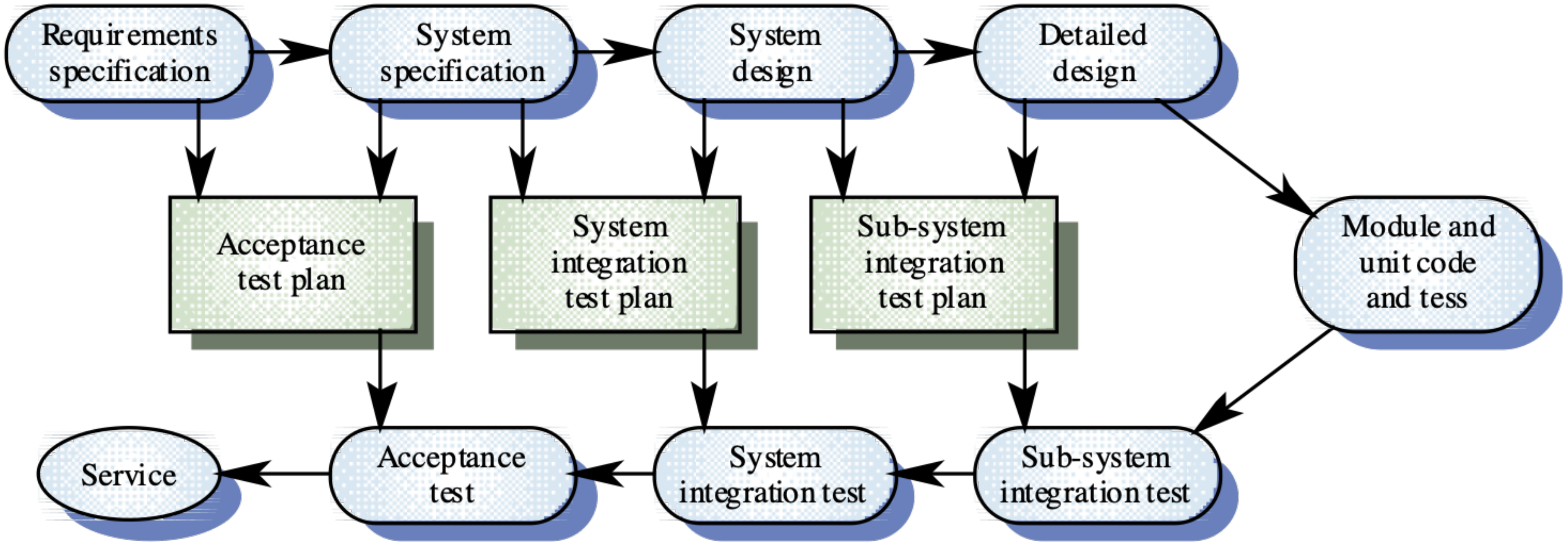
Testing stages

- **Unit testing**
  - Individual components are tested
- **Module testing**
  - Related collections of dependent components are tested
- **Sub-system testing**
  - Modules/components are integrated into sub-systems and tested. The focus here would be on interface testing
- **System testing**
  - Testing of the system as a whole. Testing of emergent properties
- **Acceptance testing**
  - Testing with customer data to check that it is acceptable

The testing process



Testing phases


Requirements specification
System specification
System design
Detailed design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and tess
Service
Acceptance test
System integration test
Sub-system integration test

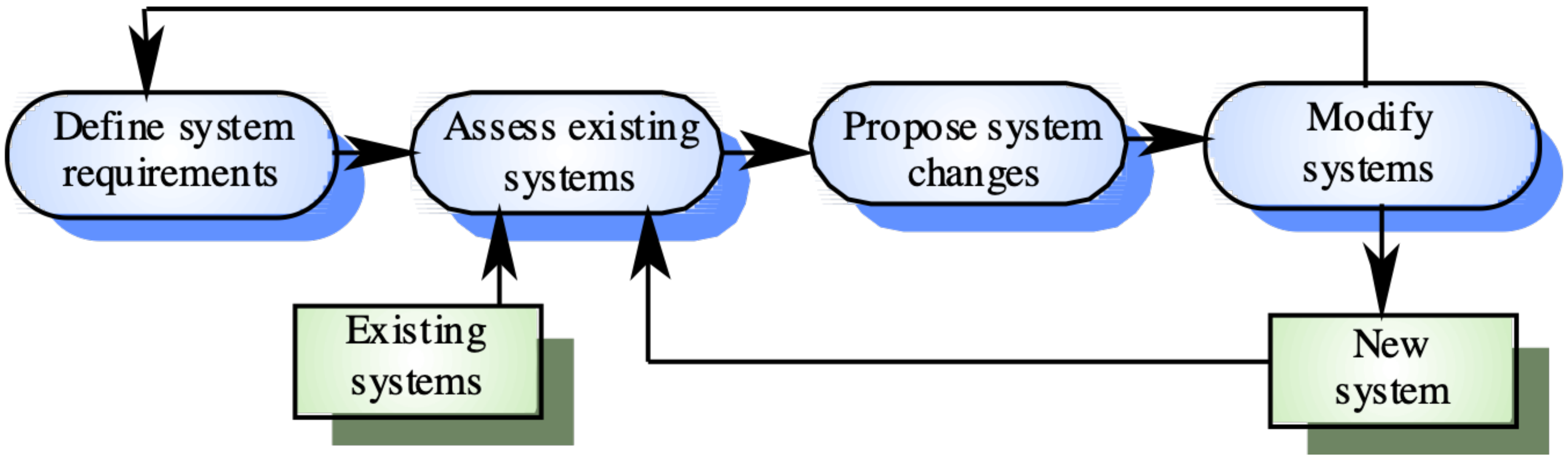
IV Software evolution

Software is inherently flexible and can change.

- As requirements change through changing business circumstances, the software that supports the business must also evolve and change

- Although there has been a demarcation between development and evolution (maintenance), this is increasingly irrelevant as fewer and fewer systems are completely new

System evolution



Summary: 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, development (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

Summary: 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