

MATA KULIAH : Rekayasa Piranti Lunak

KODE MATA KULIAH/SKS : TI1014– 4 SKS

TAHUN : 2013

VERSI : 1.0



EDUCATION FOR A BETTER LIFE



KEMAMPUAN AKHIR YANG DIHARAPKAN

Mahasiswa dapat menjelaskan
konsep dan prinsip Rekayasa
Piranti Lunak



MATERI POKOK

- Proses model Piranti Lunak
- Proses iterasi
- Aktivitas Proses
Proses terpadu
- Rekayasa Piranti Lunak dibantu komputer



Sumber Pustaka

1. A. S. Rosa, Salahuddin M., Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek, Informatika, 20013.
2. Ian Sommerville, Software Engineering 9th Edition, Addison-Wesley, 2011.
3. Roger S. Pressman, Software Engineering: A Practitioner's Approach Seventh Edition, McGraw-Hill, 2010.
4. Yasin Verdi, Rekayasa Perangkat Lunak Berorientasi Objek, Mitra Wacana Media, 2012.



The software process

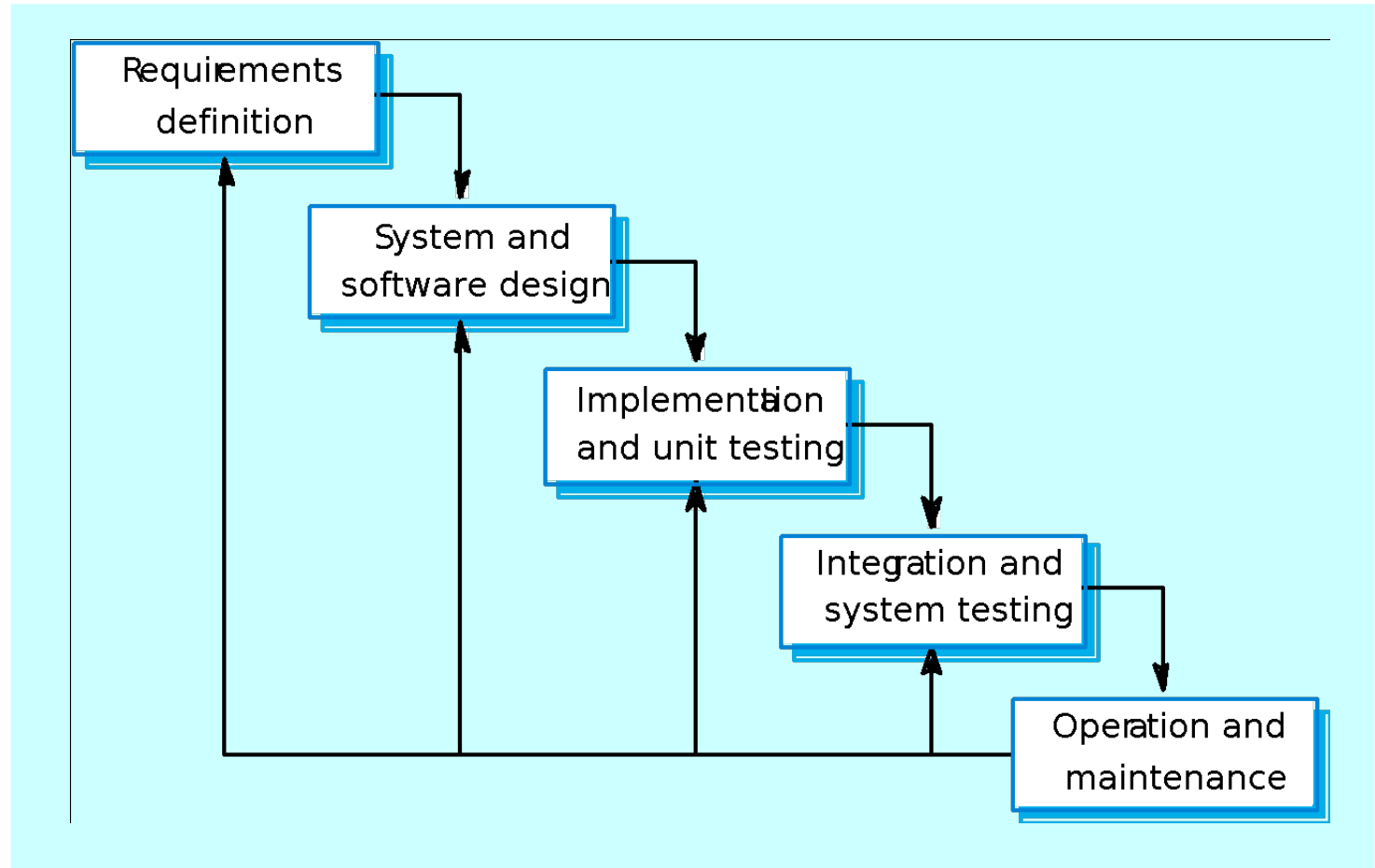
- A structured set of activities required to develop a software system
 - Specification;
 - Design;
 - Validation;
 - Evolution.
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

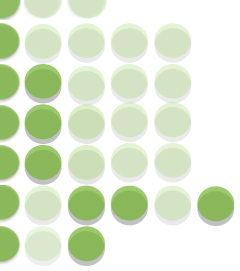


Generic software process models

- The waterfall model
 - Separate and distinct phases of specification and development.
- Evolutionary development
 - Specification, development and validation are interleaved.
- Component-based software engineering
 - The system is assembled from existing components.
- There are many variants of these models e.g. formal development where a waterfall-like process is used but the specification is a formal specification that is refined through several stages to an implementable design.

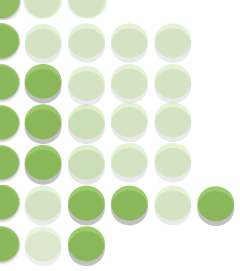
Waterfall model





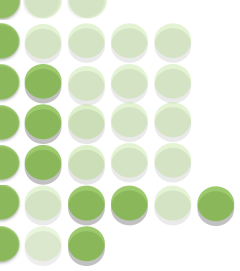
Waterfall model phases

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. One phase has to be complete before moving onto the next phase.



Waterfall model problems

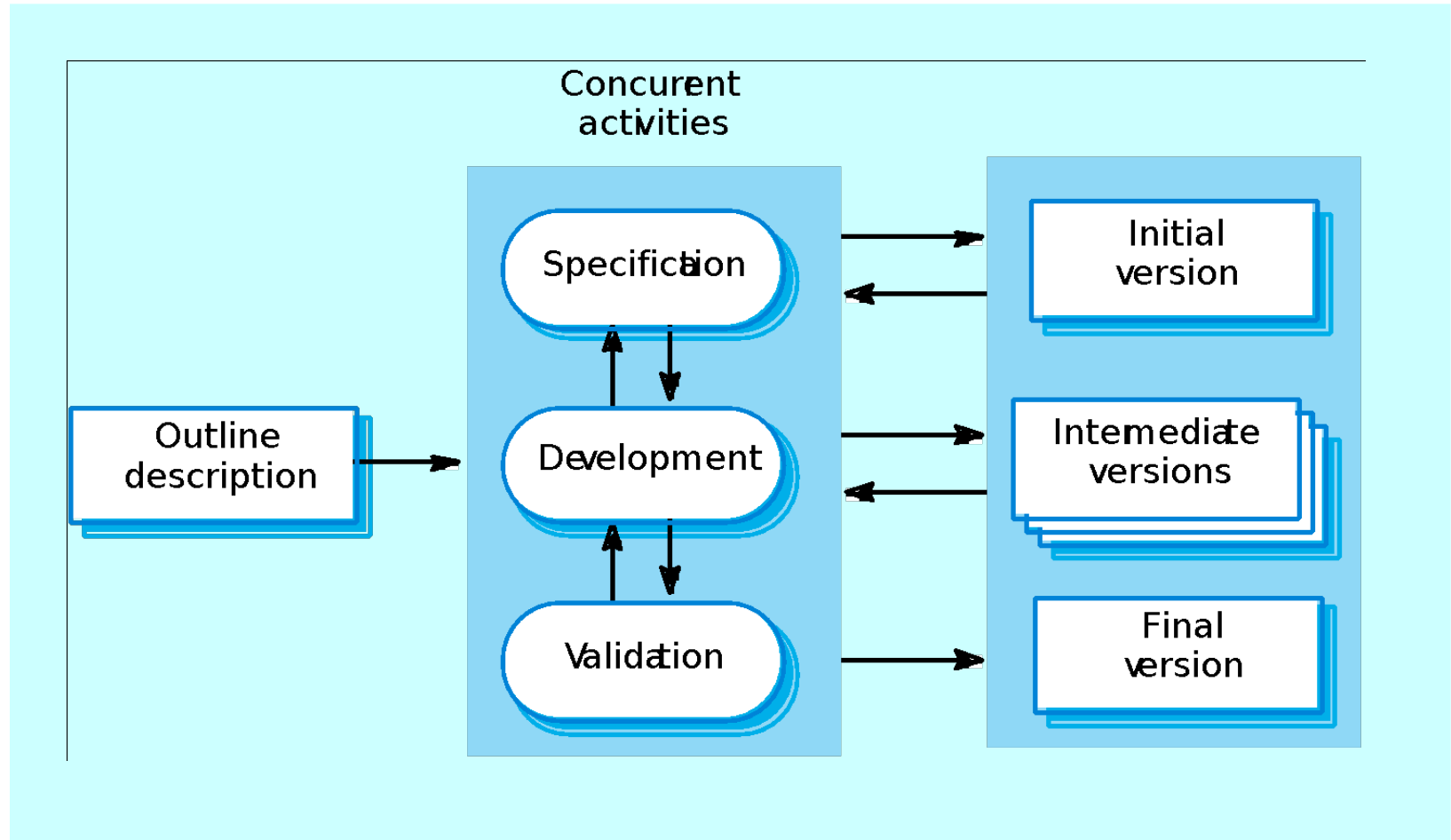
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
- Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

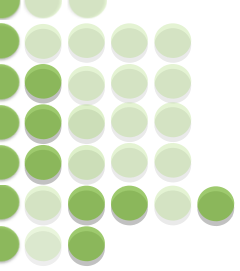


Evolutionary development

- 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 and add new features as proposed by the customer.
- Throw-away prototyping
 - Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.

Evolutionary development





Evolutionary development

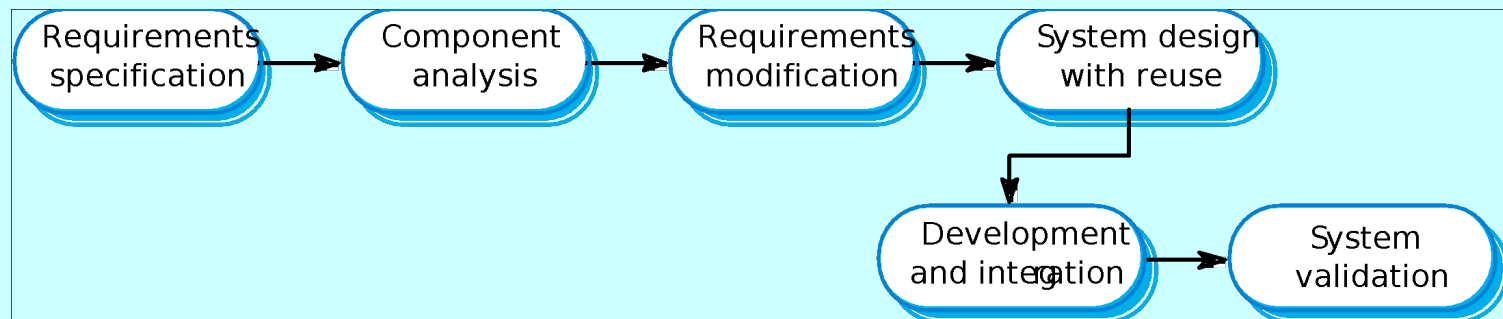
- Problems
 - Lack of process visibility;
 - Systems are often poorly structured;
 - Special skills (e.g. in languages for rapid prototyping) may be required.
- Applicability
 - For small or medium-size interactive systems;
 - For parts of large systems (e.g. the user interface);
 - For short-lifetime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

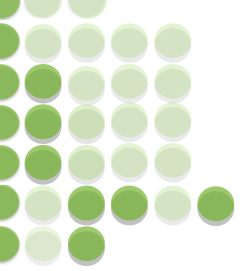


Component-based software engineering

- 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 increasingly used as component standards have emerged.

Reuse-oriented development





Process iteration

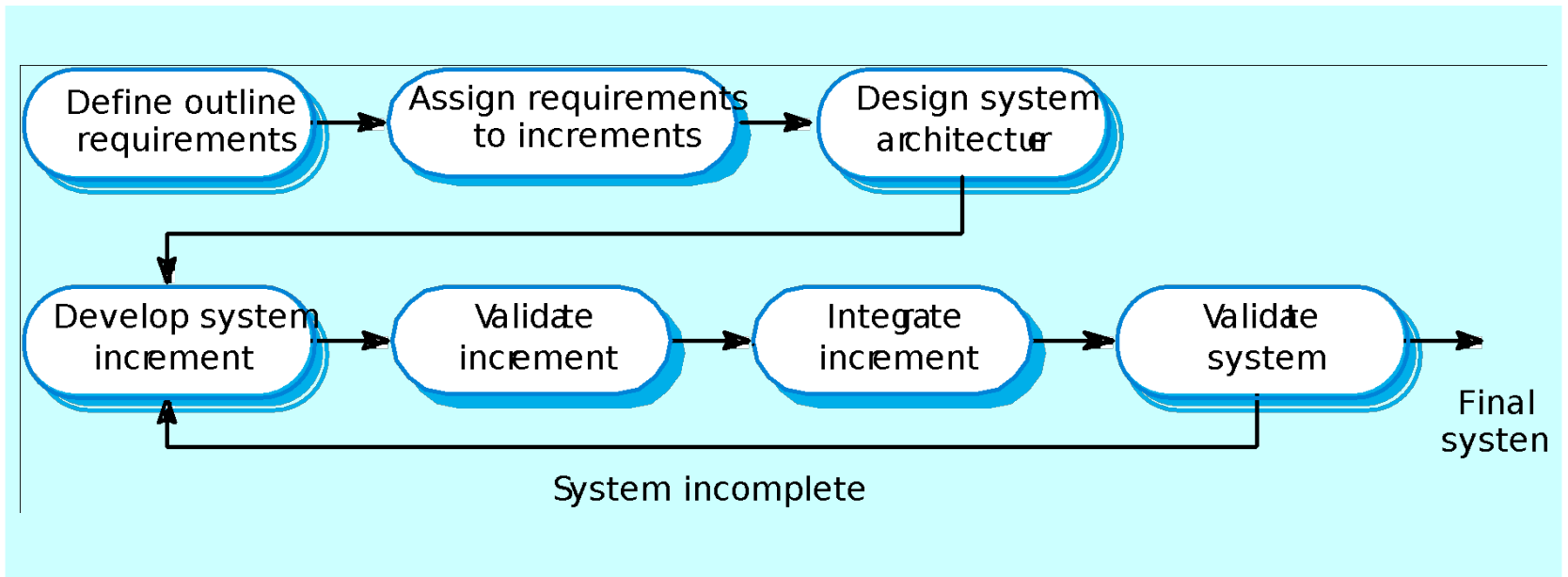
- System requirements ALWAYS evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems.
- Iteration can be applied to any of the generic process models.
- Two (related) approaches
 - Incremental delivery;
 - Spiral development.

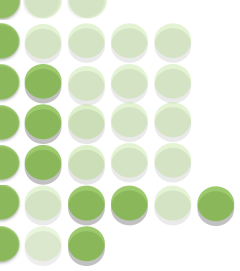


Incremental delivery

- 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





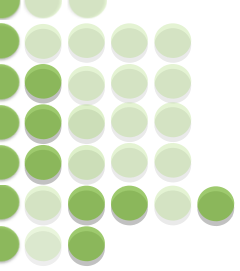
Incremental development advantages

- 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

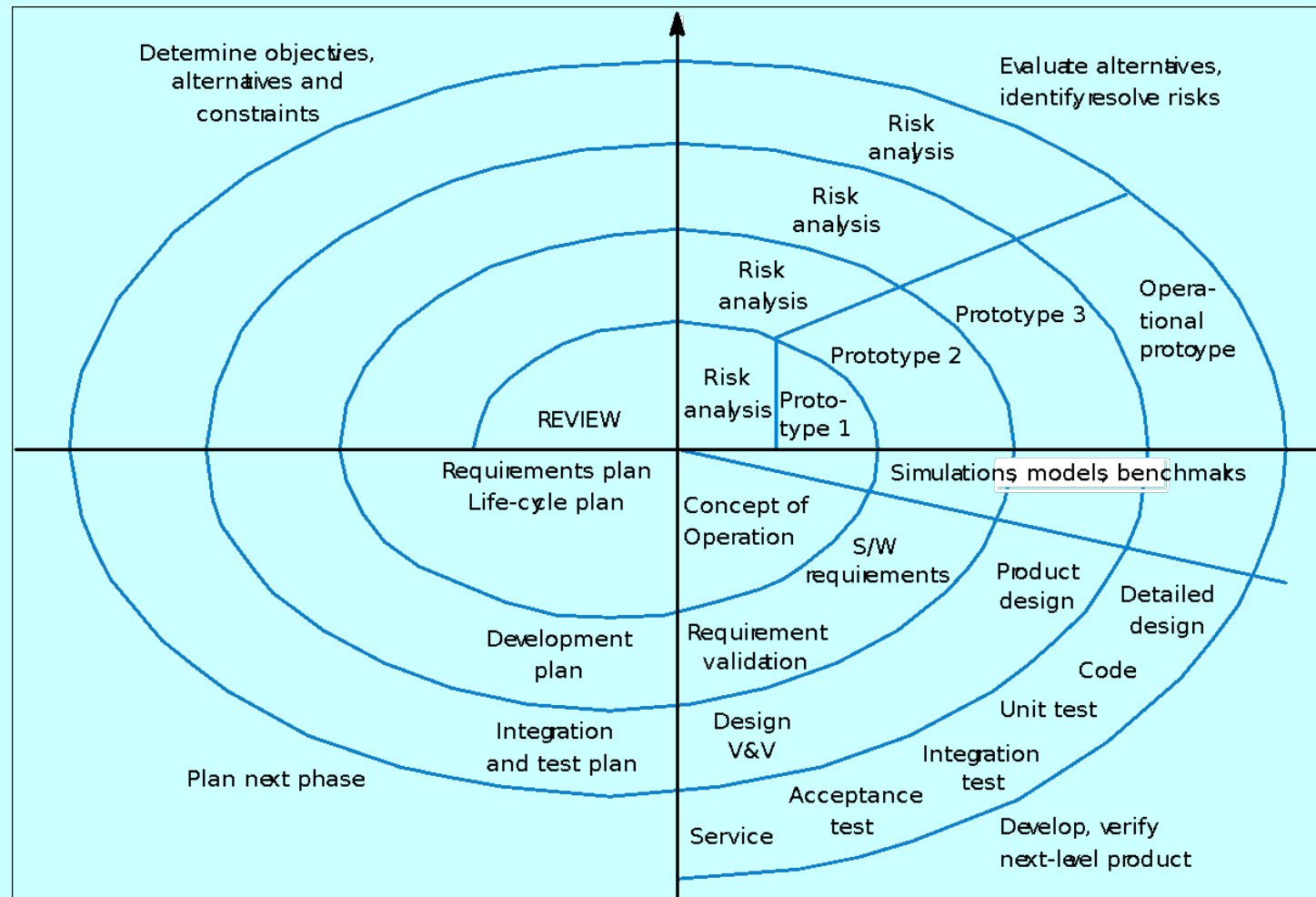
- An approach to development based on the development and delivery of very small increments of functionality.
- Relies on constant code improvement, user involvement in the development team and pairwise programming.
- Covered in Chapter 17



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.

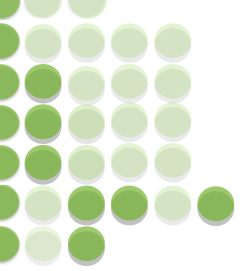
Spiral model of the software process





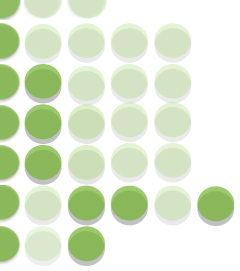
Spiral model sectors

- 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.



Key points

- Software processes are the activities involved in producing and evolving a software system.
- Software process models are abstract representations of these processes.
- General activities are specification, design and implementation, validation and evolution.
- Generic process models describe the organisation of software processes. Examples include the waterfall model, evolutionary development and component-based software engineering.
- Iterative process models describe the software process as a cycle of activities.



Process activities

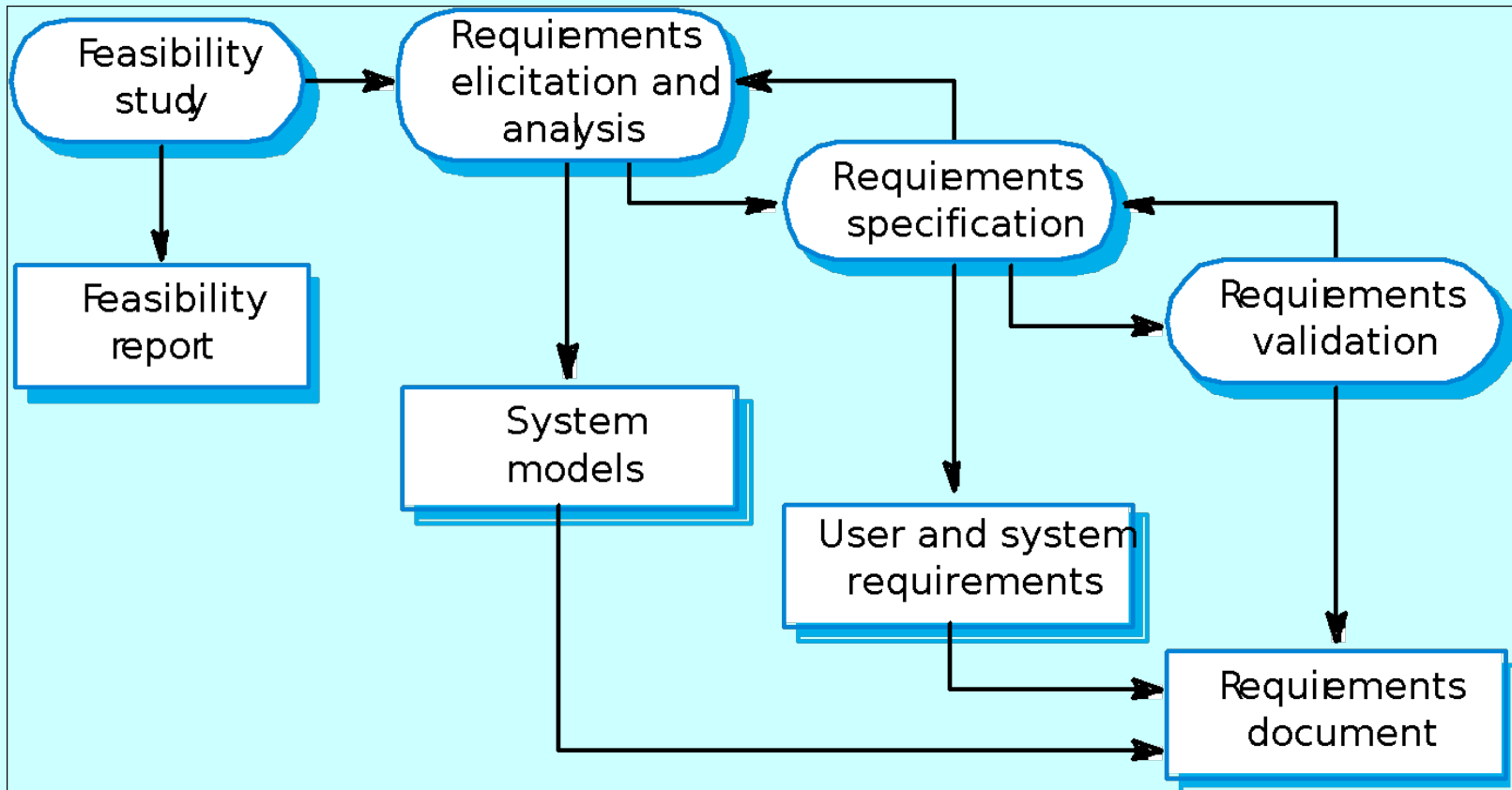
- Software specification
- Software design and implementation
- Software validation
- Software evolution



Software specification

- The process of establishing what services 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.

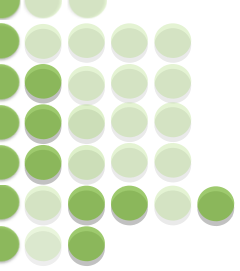
The requirements engineering process





Software design and implementation

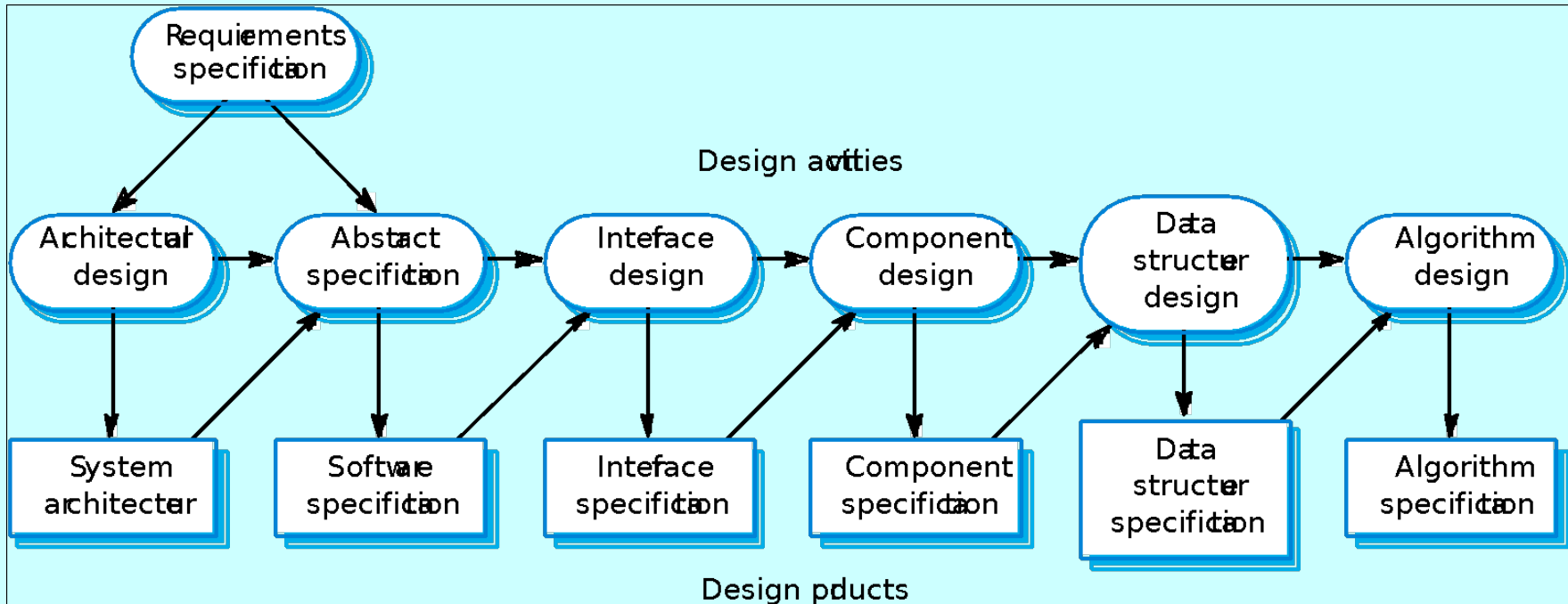
- The process of converting the system specification into an executable system.
- Software design
 - Design a software structure that realises the specification;
- Implementation
 - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be inter-leaved.



Design process activities

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

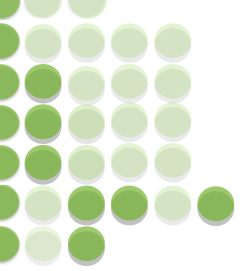
The software design process





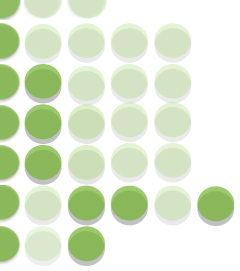
Structured methods

- Systematic approaches to developing a software design.
- The design is usually documented as a set of graphical models.
- Possible models
 - Object model;
 - Sequence model;
 - State transition model;
 - Structural model;
 - Data-flow model.



Programming and debugging

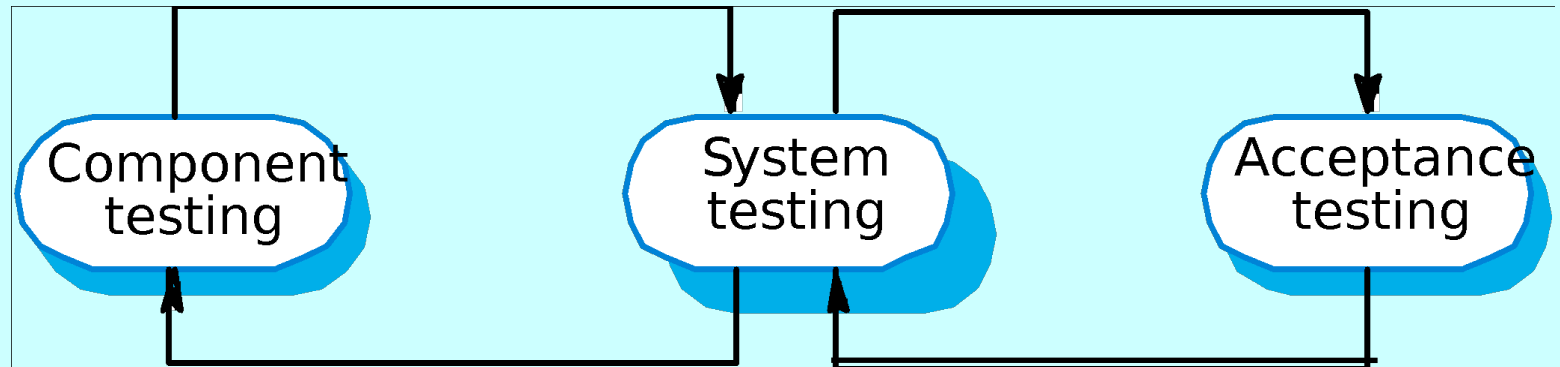
- Translating a design into a program and removing errors from that program.
- Programming is a personal activity - there is no generic programming process.
- Programmers carry out some program testing to discover faults in the program and remove these faults in the debugging process.



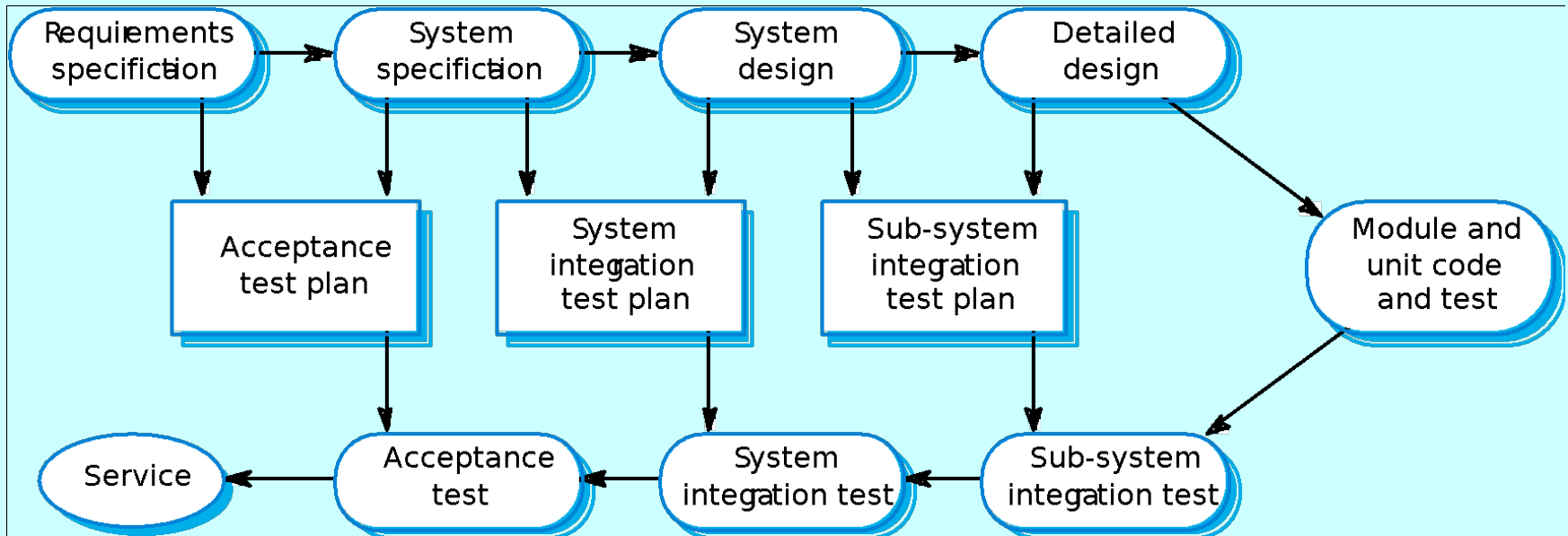
Software validation

- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system 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.

The testing process



Testing phases

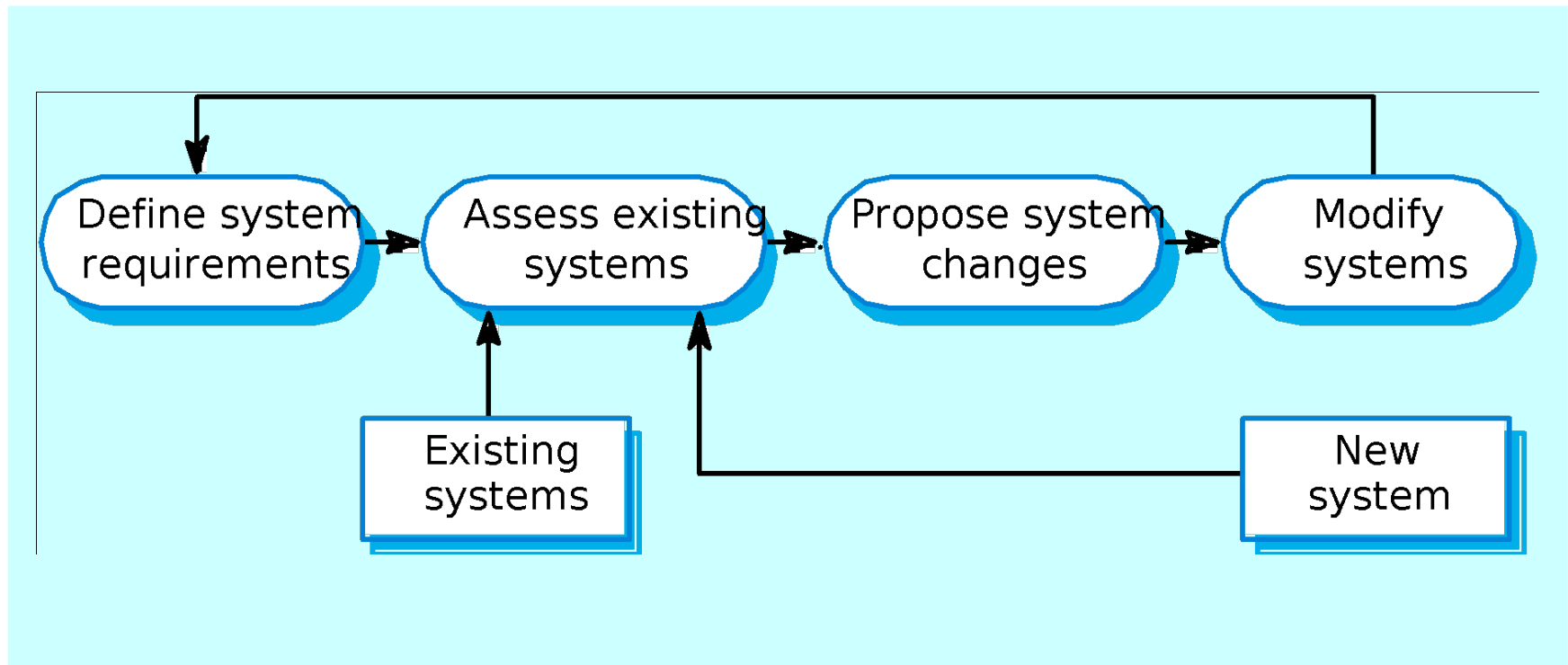


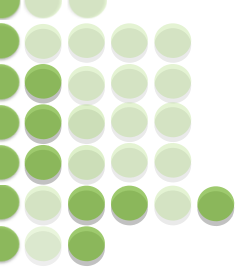


Software evolution

- 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

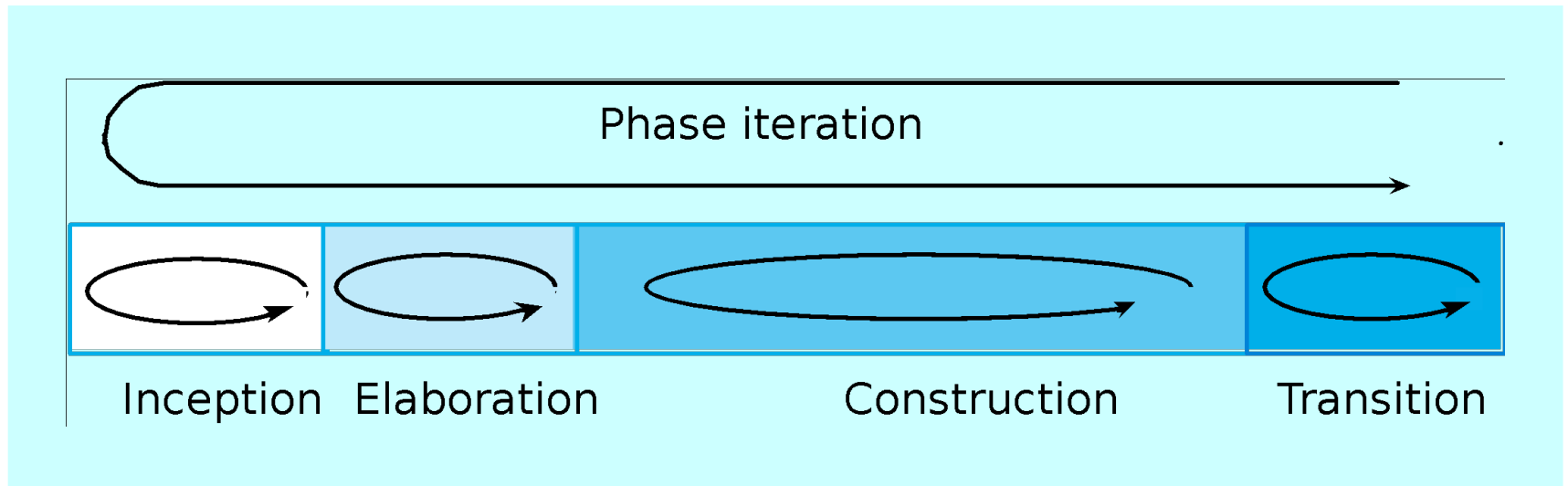


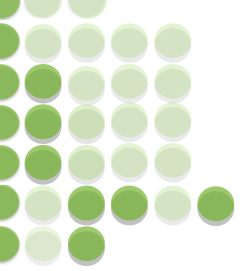


The Rational Unified Process

- A modern process model derived from the work on the UML and associated process.
- Normally described from 3 perspectives
 - A dynamic perspective that shows phases over time;
 - A static perspective that shows process activities;
 - A practice perspective that suggests good practice.

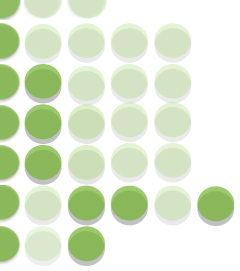
RUP phase model





RUP phases

- Inception
 - Establish the business case for the system.
- Elaboration
 - Develop an understanding of the problem domain and the system architecture.
- Construction
 - System design, programming and testing.
- Transition
 - Deploy the system in its operating environment.



RUP good practice

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software



Static workflows

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
Test	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace
Configuration and change management	This supporting workflow manages changes to the system (see Chapter 29)
Project management	This supporting workflow manages the system development (see Chapter 5)
Environment	This workflow is concerned with making appropriate software tools available to the software development team.



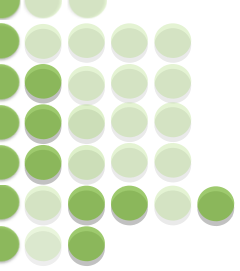
Computer-aided software engineering

- 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. However, these are not the order of magnitude improvements that were once predicted
 - Software engineering requires creative thought - this is not readily automated;
 - Software engineering is a team activity and, for large projects, much time is spent in team interactions. CASE technology does not really support these.

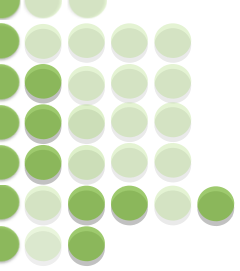


CASE classification

- Classification helps us understand the different types of CASE tools and their support for process activities.
- Functional perspective
 - Tools are classified according to their specific function.
- Process perspective
 - Tools are classified according to process activities that are supported.
- Integration perspective
 - Tools are classified according to their organisation into integrated units.

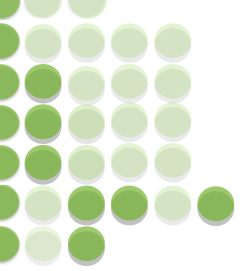
Activity-based tool classification

Re-engineering tools			•	
Testing tools			•	•
Debugging tools			•	•
Program analysis tools			•	•
Language-processing tools		•	•	
Method support tools	•	•		
Prototyping tools	•			•
Configuration management tools		•	•	
Change management tools	•	•	•	•
Documentation tools	•	•	•	•
Editing tools	•	•	•	•
Planning tools	•	•	•	•
	Specification	Design	Implementation	Verification and Validation



Summary

- 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.
- The Rational Unified Process is a generic process model that separates activities from phases.
- CASE technology supports software process activities.



EDUCATION FOR A BETTER LIFE



EDUCATION FOR A BETTER LIFE