SOFTWARE PROCESS

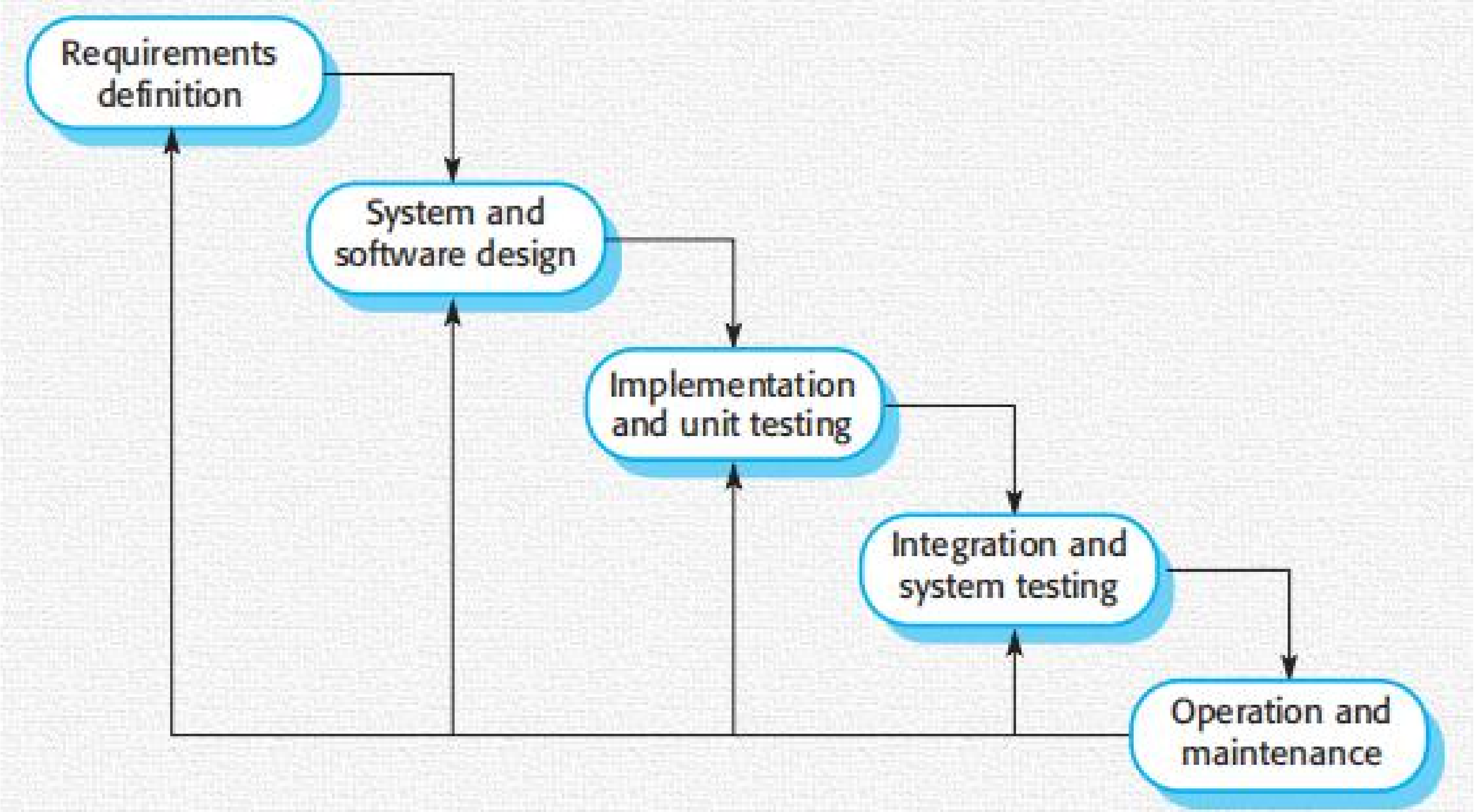
- Software Process Model
- Process Activities
- Dealing with Changes

SOME SOFTWARE PROCESS MODELS

- The Waterfall model
  - Plan-driven model
  - Separate and distinct phases of specification and development
- Evolutionary/ Incremental Development
  - Specification, development and validation are interleaved
  - Might be plan-driven or agile
- Component-based/ Integration and configuration
  - The system is assembled from existing configurable components

Note: In practice, most large systems are developed using a process that incorporates elements from all of these models

The Waterfall model


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

In principle, a phase has to be complete before moving onto the next phase

The Waterfall model

The main drawback:

- the difficulty of accommodating change after the process is underway.

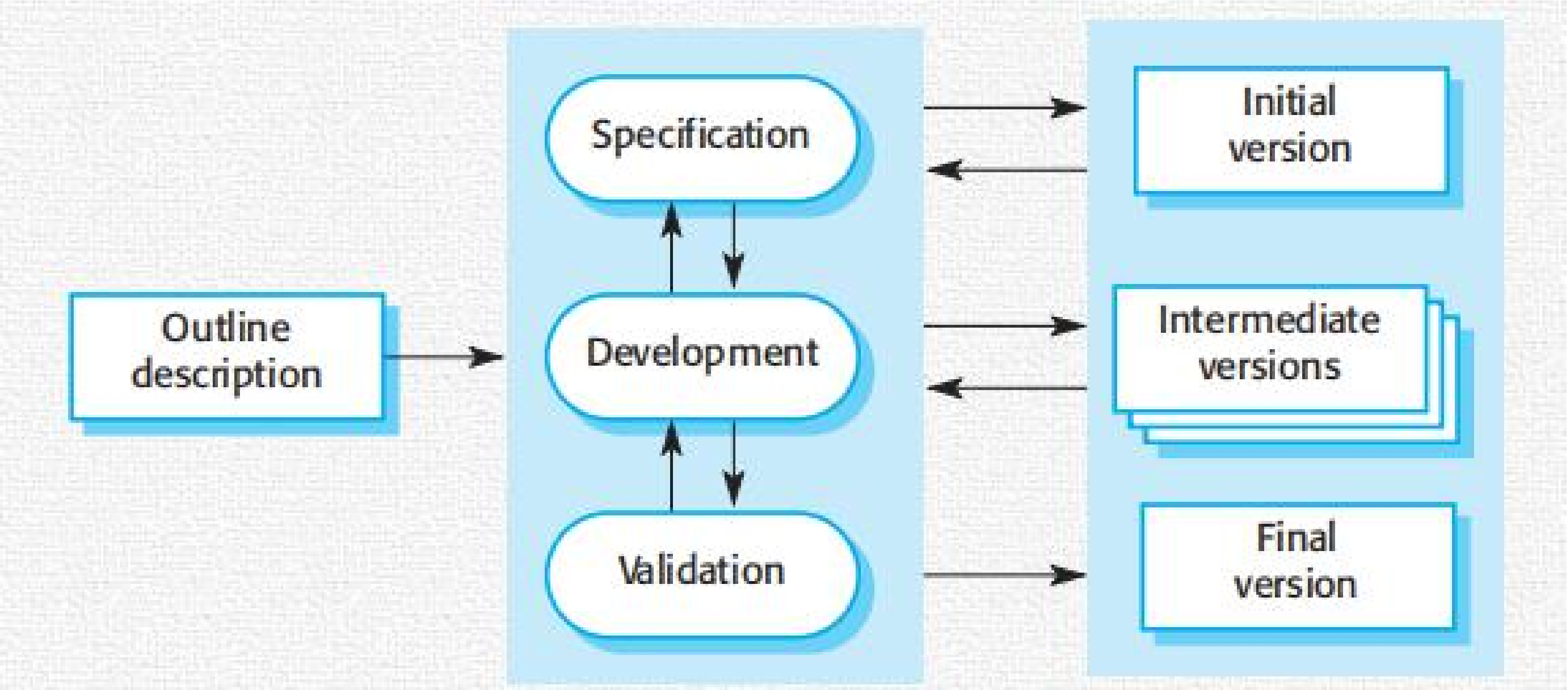
When the requirements are well-understood and changes will be fairly limited during the design process.

- Few business systems have stable requirements.

Mostly used for large systems engineering projects

- a system is developed at several sites.
- the plan-driven nature of the waterfall model helps coordinate the work.

Evolutionary/ Incremental Development



Concurrent activities

Evolutionary/ Incremental Development

Incremental development benefits

- Reduce the cost of accommodating changing customer requirements
- Easier to get customer feedback on the development work that has been done.
- More rapid delivery and deployment of useful software to the customer

Evolutionary/ Incremental Development

Incremental development problems

- The process is not visible.
- Managers need regular deliverables
- Not cost-effective to produce documents for every product version
- System structure tends to degrade as new increments are added.
- Need time and money on refactoring to improve the software
- Regular change tends to corrupt the structure.
- Incorporating further software changes becomes increasingly difficult and costly

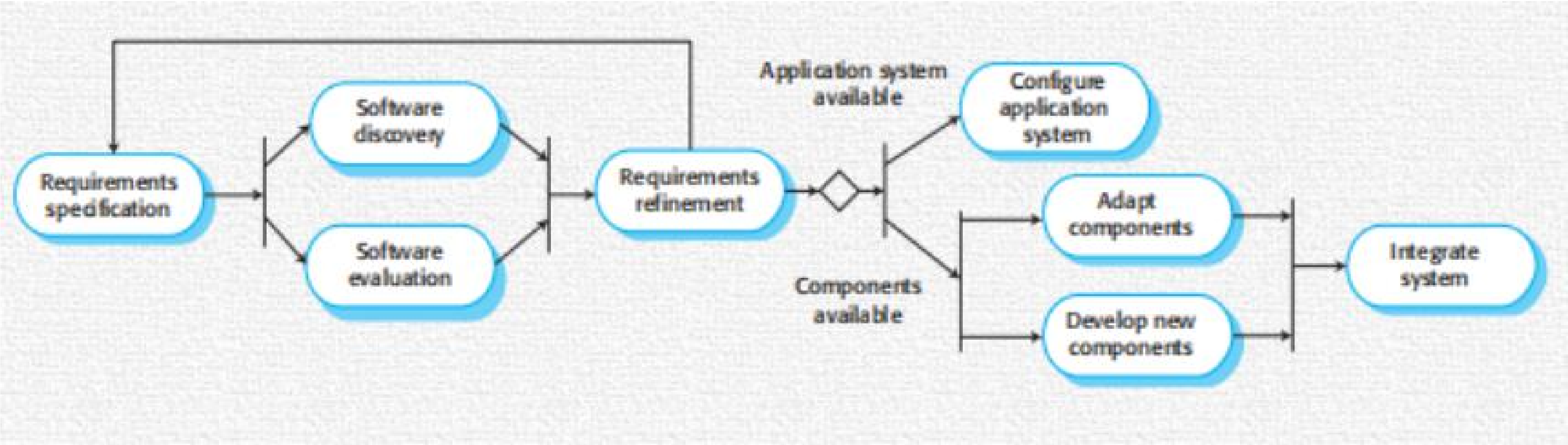
Component-based/ Integration and configuration

- Based on software reuse where systems are integrated from existing components or application systems

- Reused elements may be configured to adapt their behaviour and functionality to a user’s requirements

- Reuse is now the standard approach for building many types of business system

Component-based/ Integration and configuration



Component-based/ Integration and configuration

Advantages and disadvantages

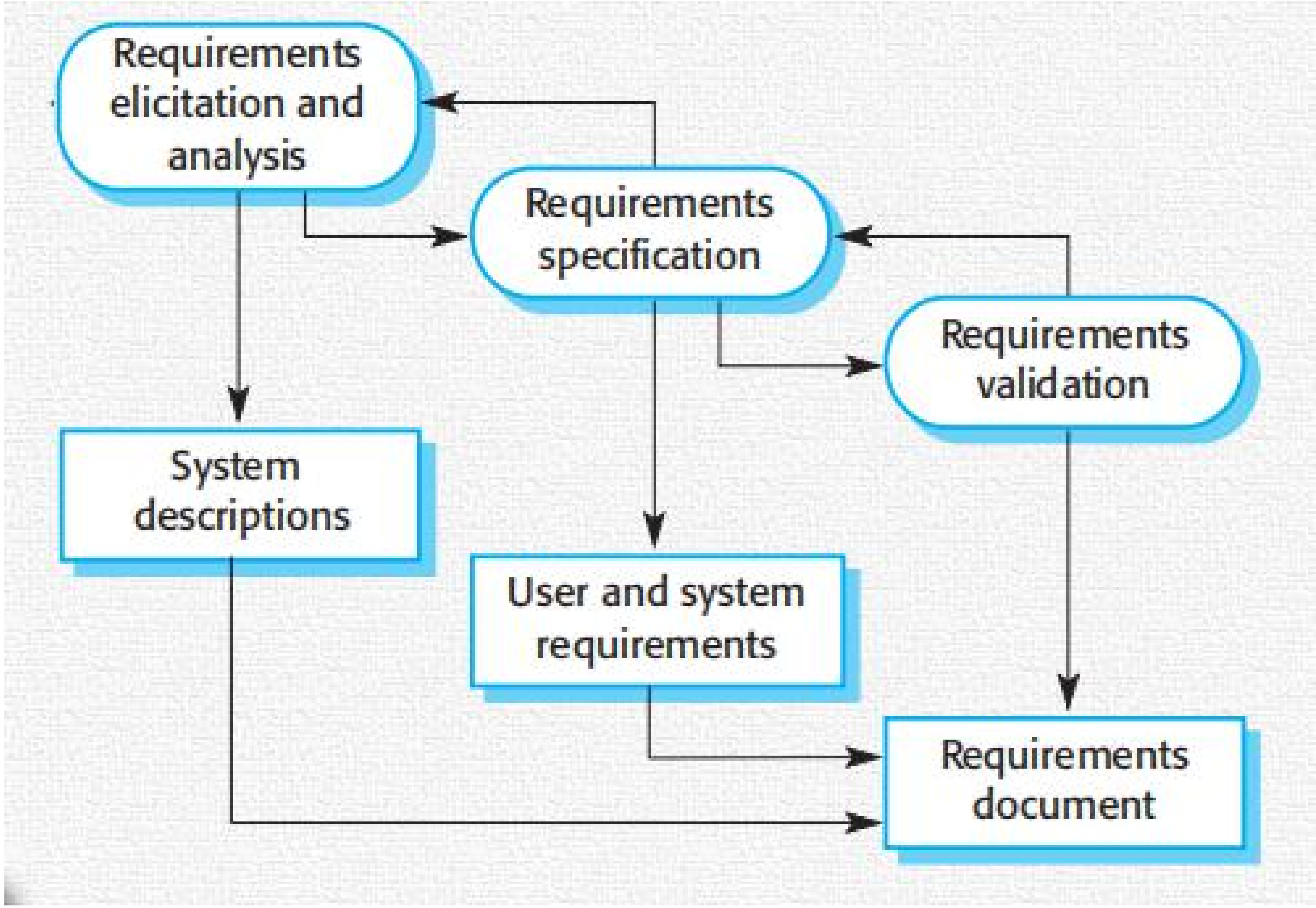
- Reduced costs and risks as less software is developed from scratch
- Faster delivery and deployment of system
- But requirements compromises are inevitable so system may not meet real needs of users
- Loss of control over evolution of reused system elements

PROCESS ACTIVITIES

Activity: Software specification

- The process of establishing what services are required and the constraints on the system’s operation and development.
- Use: Requirements engineering process
  - Requirements elicitation and analysis
  - Requirements specification
  - Requirements validation

THE ENGINEERING PROCESS


Requirements elicitation and analysis
Requirements specification
Requirements validation
System descriptions
User and system requirements
Requirements document

PROCESS ACTIVITIES

Activity: Software Design and Implementation ~ Development

٥ The process of converting the system speifications into an executable system

Two (sub) activities:

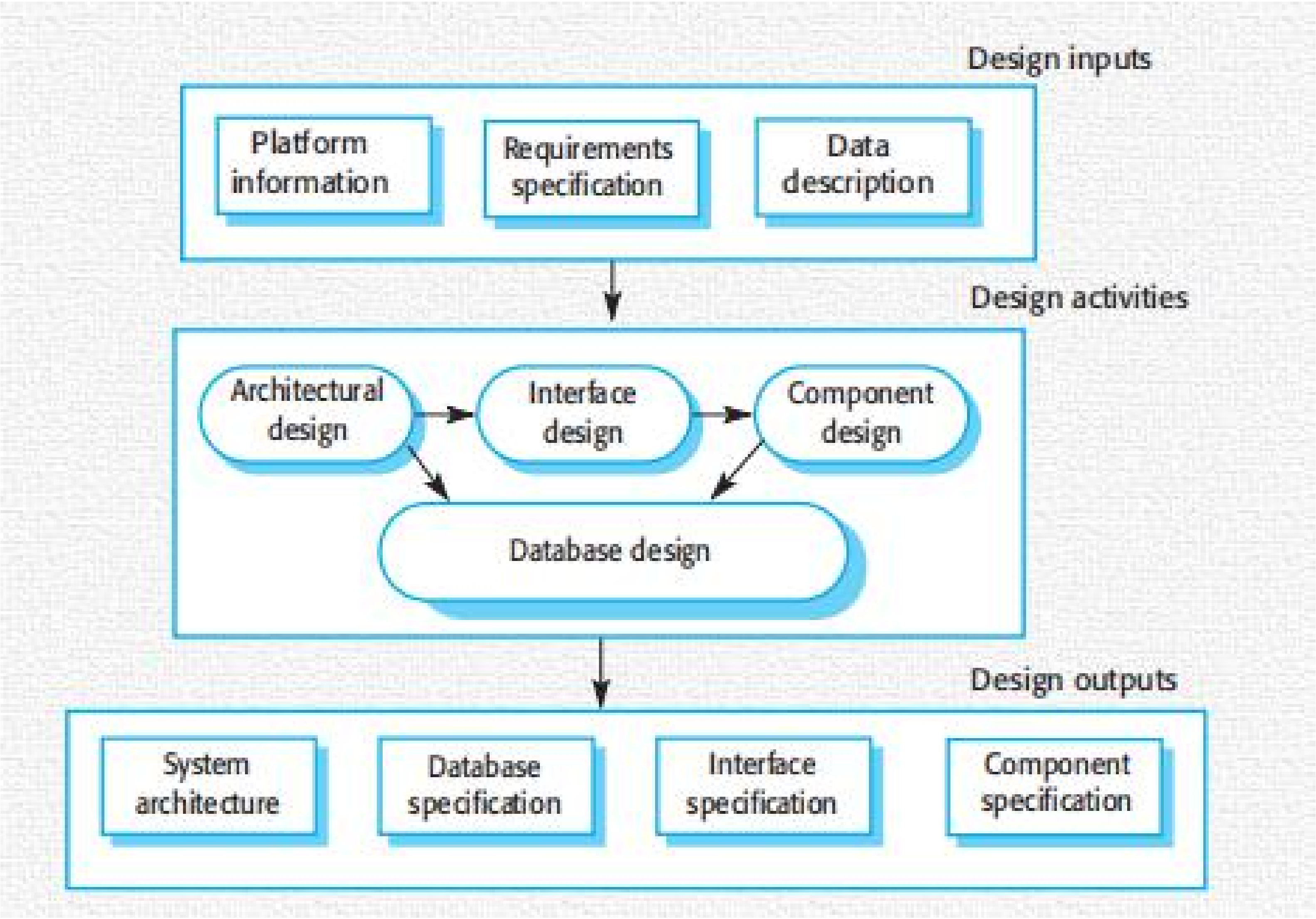
- Software design

Design a software structure that realises the specification;

- Implementation

Translate this structure into an executable program

THE ENGINEERING PROCESS FOR DESIGN



SOFTWARE IMPLEMENTATION

The software is implemented either by developing a program or programs or by configuring an application system.

• Design and implementation are interleaved activities for most types of software system.
 i.e: For waterfall model, implementation is taken after design. In other software process models, design and implementation can be *interleaved*

• Programming is an individual activity with no standard process.
• Debugging is the activity of finding program faults and correcting these faults.

PROCESS ACTIVITIES

Activity: Software Validation

Verification and validation (V & V)

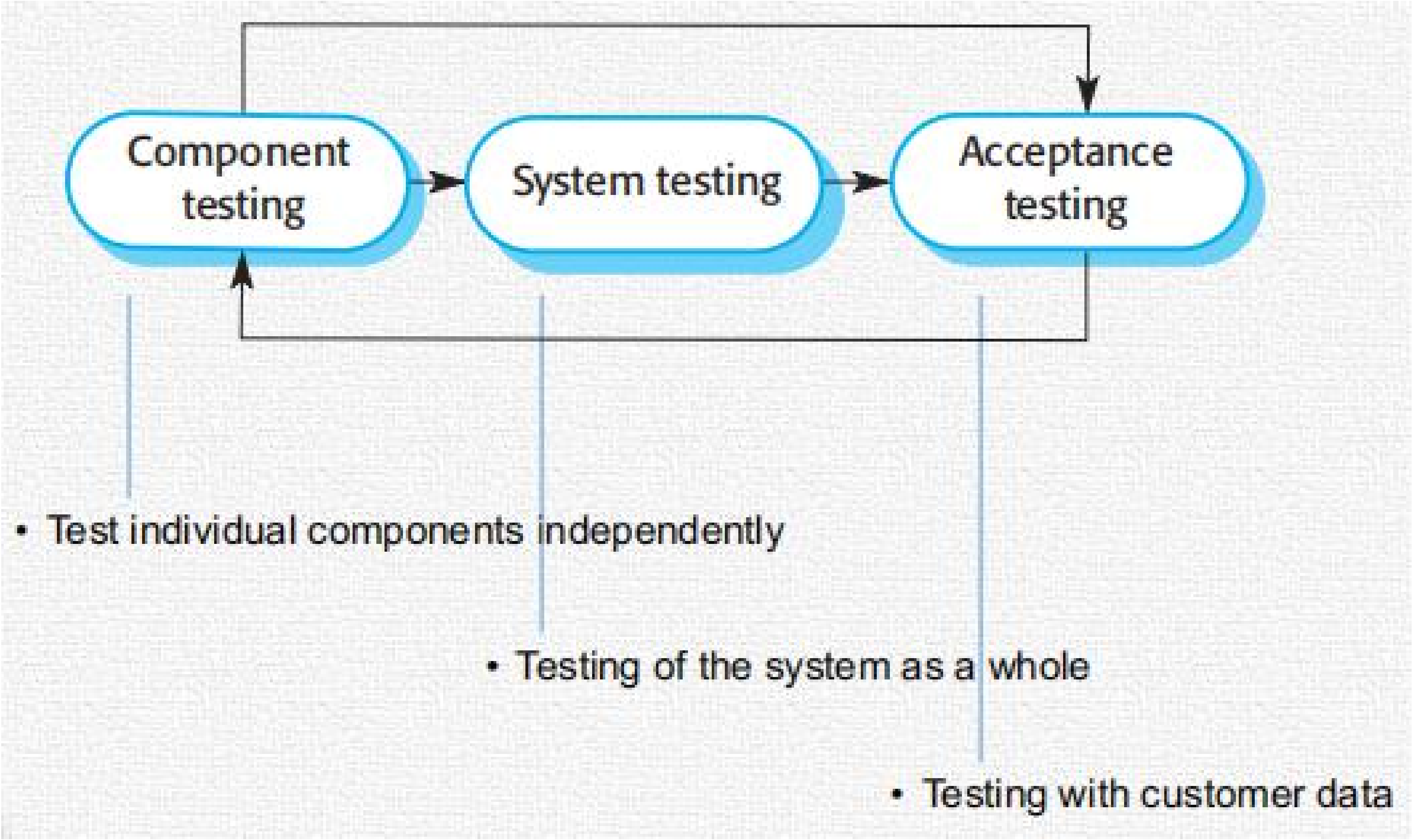
- to show that a system conforms to its specification and meets the requirements of the system customer

Involves checking and review processes and system testing.

- System testing: executing the system with test cases
- Testing: the most commonly used V & V activity

VERIFICATION vs VALIDATION

THE ENGINEERING STAGE OF TESTING



- Test individual components independently

- Testing of the system as a whole

- Testing with customer data

PROCESS ACTIVITIES

Activity: Software Evolution

Software is inherently flexible and can change.

- Requirements can change
- (changing business circumstances) => the software must also evolve and change.

<u>Dealing with change?</u>

SOFTWARE PROCESS

- Software Process Model
- Process Activities
- Dealing with Changes

DEALING WITH CHANGE

Change is inevitable in all large software projects.

- Business changes
- New technologies
- Changing platforms

Change leads to rework

- costs include rework (re-analysing requirements) and implementing new functionality

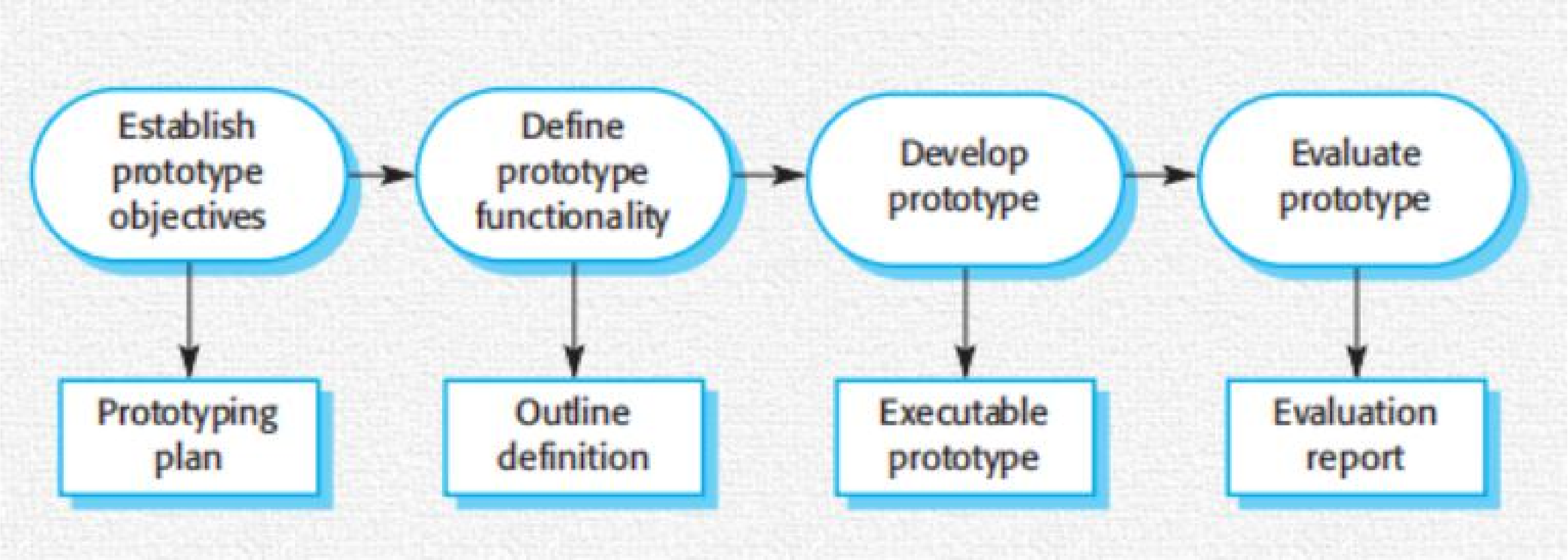
Software prototyping

A prototype is an initial version of a system used to demonstrate concepts and try out design options.

A prototype can be used in:

- requirements engineering process: requirements elicitation and validation;
- design processes: options and develop UI design;
- testing process: run back-to-back tests.

Prototype Development



A popular term: **MVP**

- a version of a product with just enough features to be usable by early customers who can then provide feedback for future development

DEALING WITH CHANGE

Incremental delivery

The development and delivery is broken down into increments

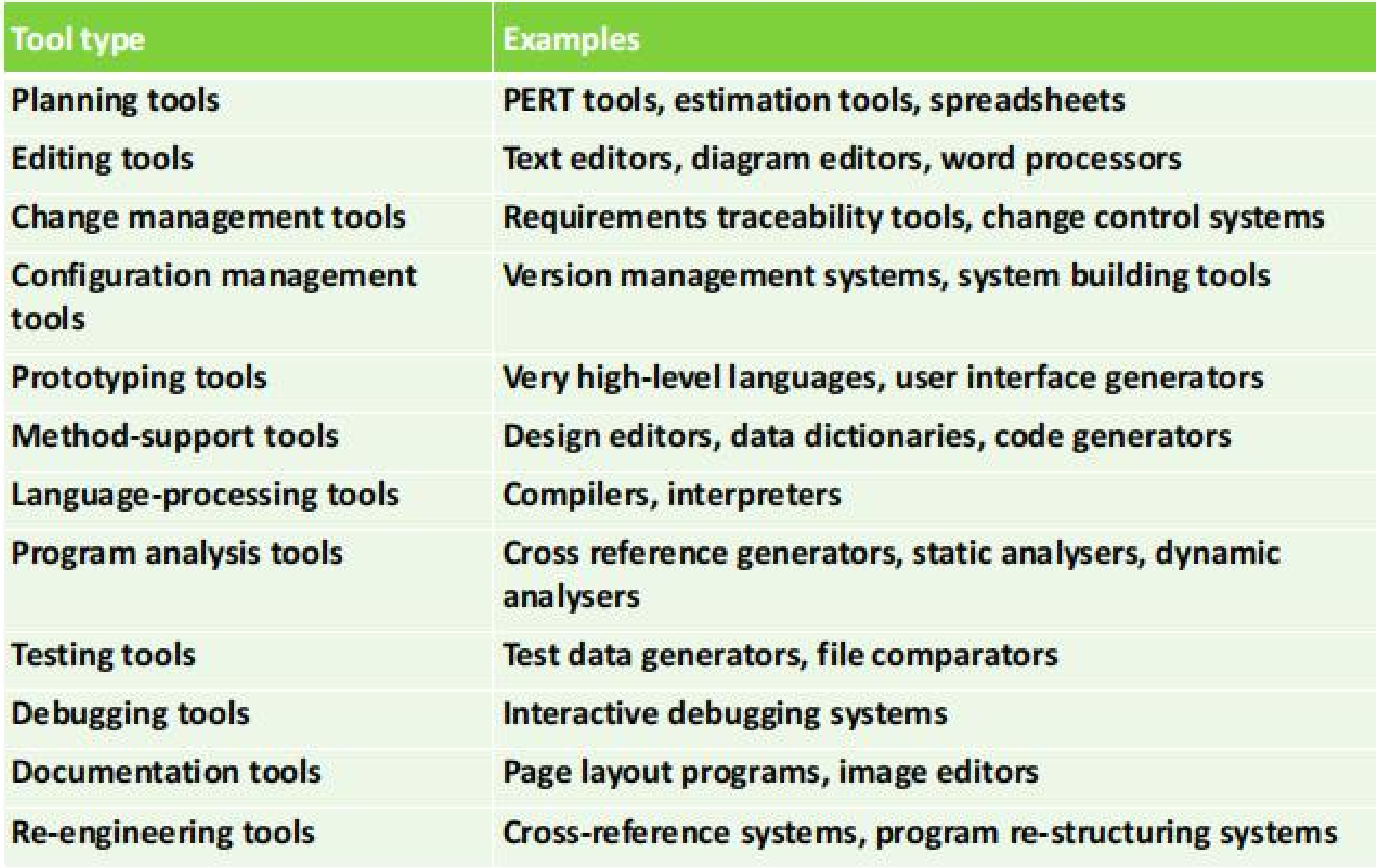
- each increment delivering part of the required functionality.
- user requirements are prioritised and the highest priority requirements are included in early increments

CASE (Computer-aided software engineering)

Computer-aided software engineering (CASE) is software to support software development and evolution processes.

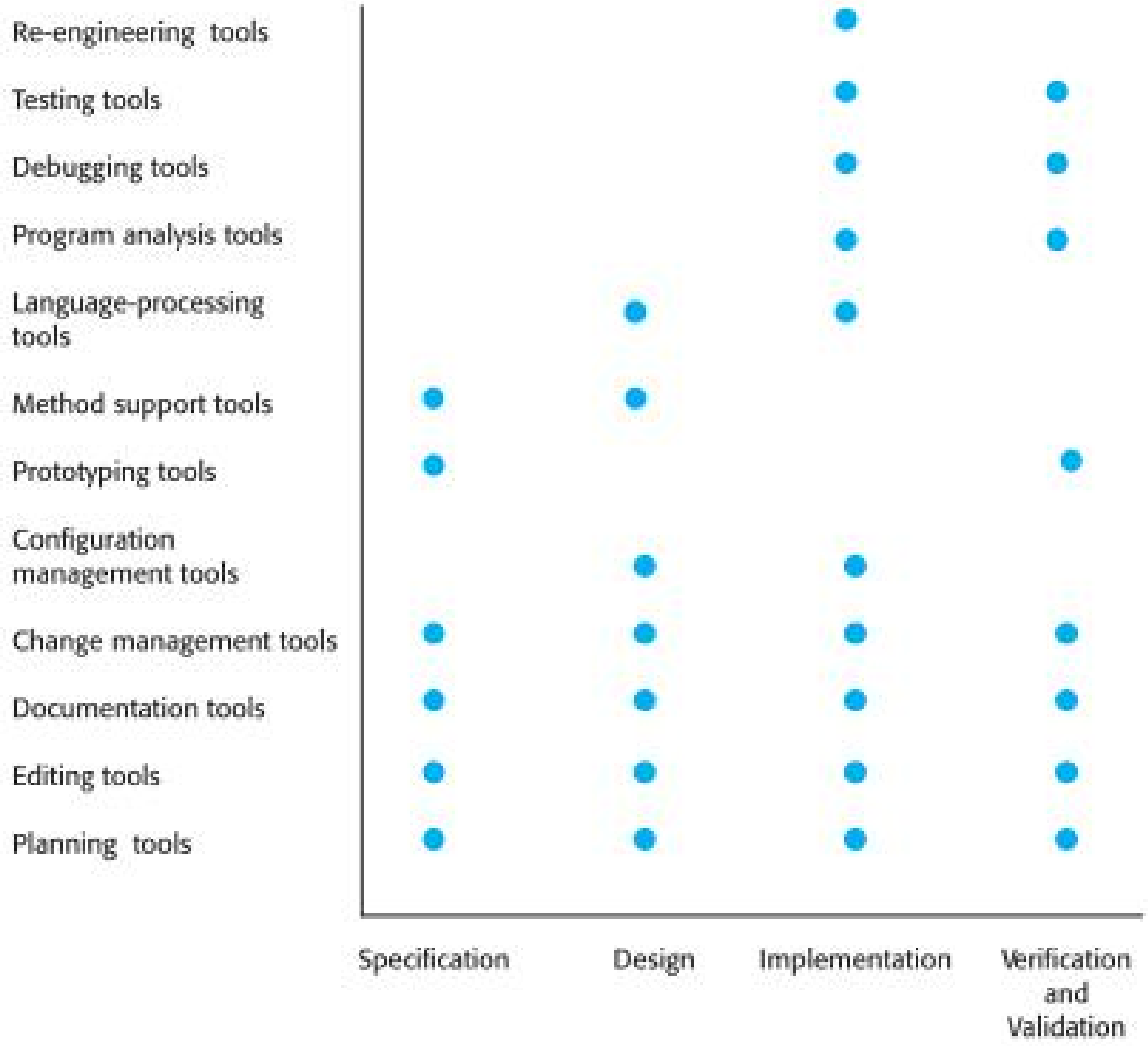
Activity automation
– Graphical editors for system model development;
– Data dictionary to manage design entities;
– Graphical UI builder for user interface construction;
– Debuggers to support program fault finding;
– Automated translators to generate new versions of a program.

Functional CASE tools



Tool type	Examples
Planning tools	PERT tools, estimation tools, spreadsheets
Editing tools	Text editors, diagram editors, word processors
Change management tools	Requirements traceability tools, change control systems
Configuration management tools	Version management systems, system building tools
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

Activitiy-based CASE tools



	Specification	Design	Implementation	Verification and Validation
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	●	●	●	●

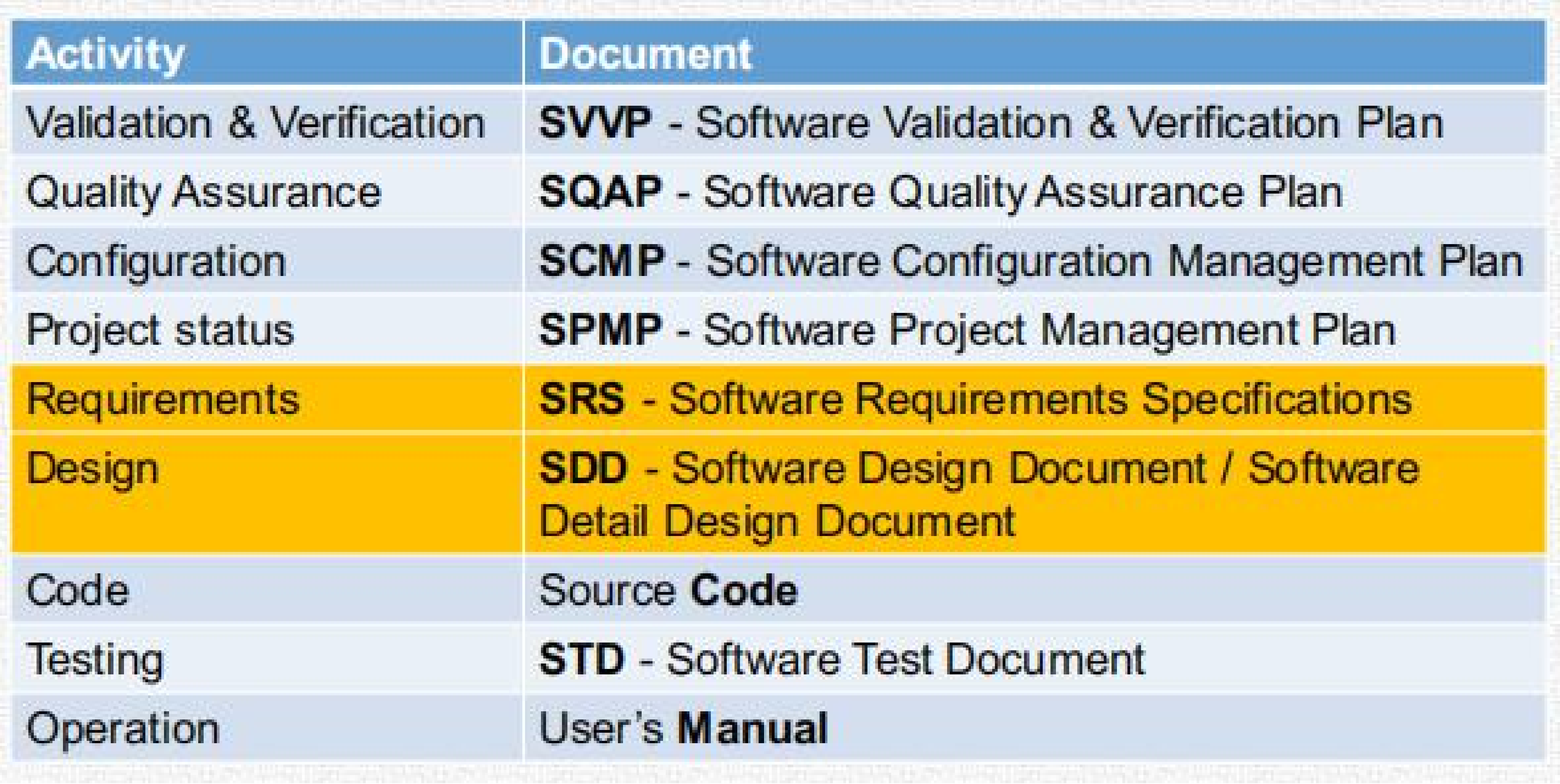
KEY POINTS

- 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.

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.
- Incremental delivery break down the development process into increments
- CASE technology supports software process activities.

Software Project Documentation



Activity	Document
Validation & Verification	**SVVP** - Software Validation & Verification Plan
Quality Assurance	**SQAP** - Software Quality Assurance Plan
Configuration	**SCMP** - Software Configuration Management Plan
Project status	**SPMP** - Software Project Management Plan
Requirements	**SRS** - Software Requirements Specifications
Design	**SDD** - Software Design Document / Software Detail Design Document
Code	Source **Code**
Testing	**STD** - Software Test Document
Operation	User's **Manual**