UNIT-I

Overview: Introduction:
Professional Software Development, Software Engineering Ethics, Case studies.

Software Processes: Models, Process activities, Coping with Change, Process improvement.

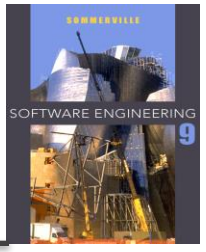
Requirements Engineering and System Modelling:
Software Requirements: Functional and Non-functional requirements, Requirements Elicitation, Specification, Validation and Change

Topics covered

Software Processes

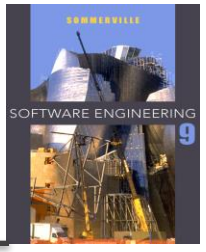
- ✧ Software process models
- ✧ Process activities
- ✧ Coping with change
- ✧ Process improvement

The software process



- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

Software process descriptions



- When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
- Process descriptions may also include:
  - Products, which are the outcomes of a process activity;
  - Roles, which reflect the responsibilities of the people involved in the process;
  - Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

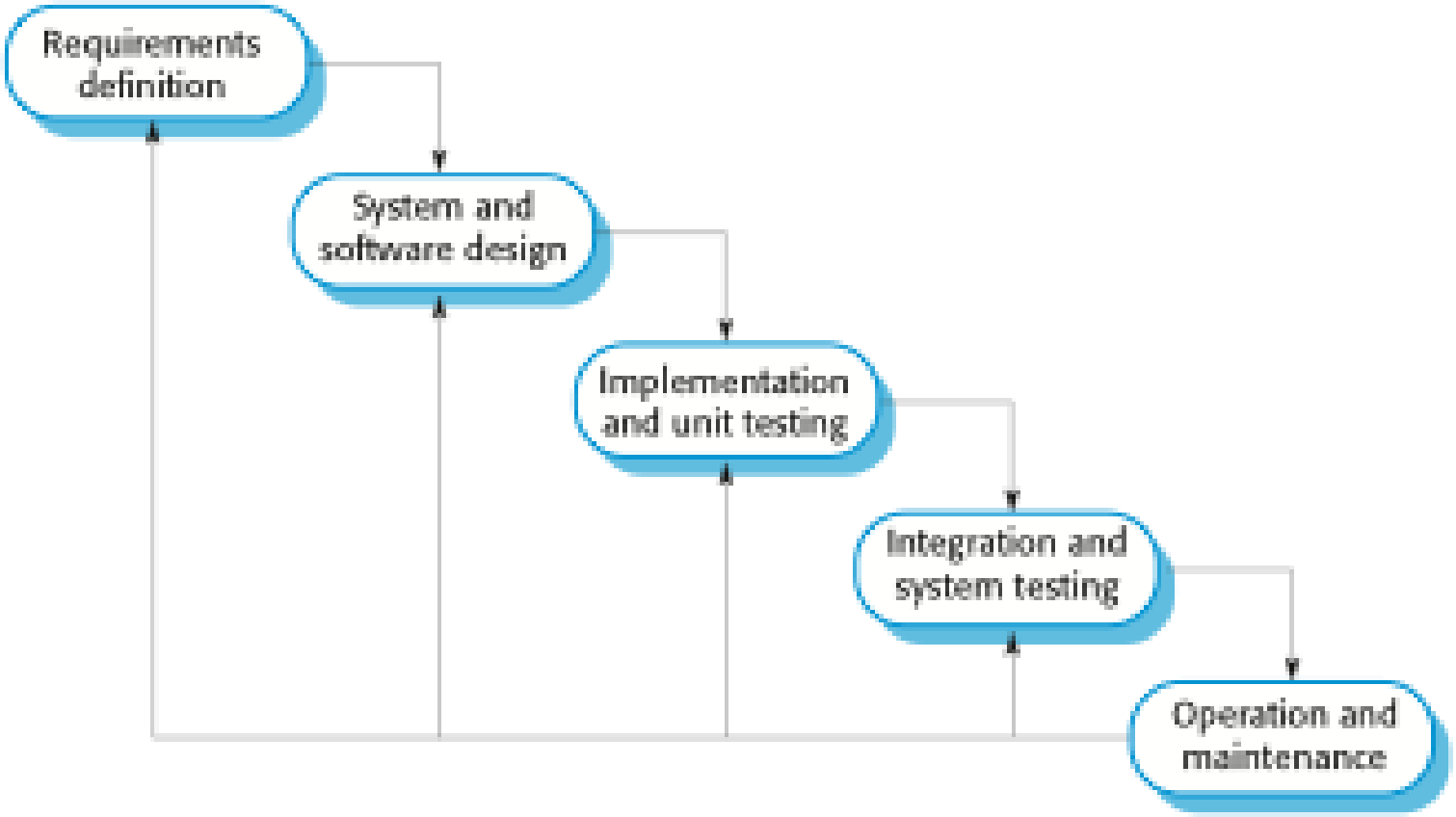
Plan-driven and agile processes

- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

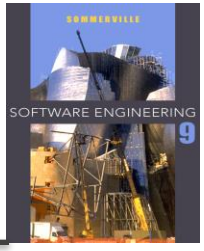
Software process models

- The waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
- Incremental development
  - Specification, development and validation are interleaved. May be plan-driven or agile.
- Reuse-oriented software engineering
  - The system is assembled from existing components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

The waterfall model



Waterfall model phases

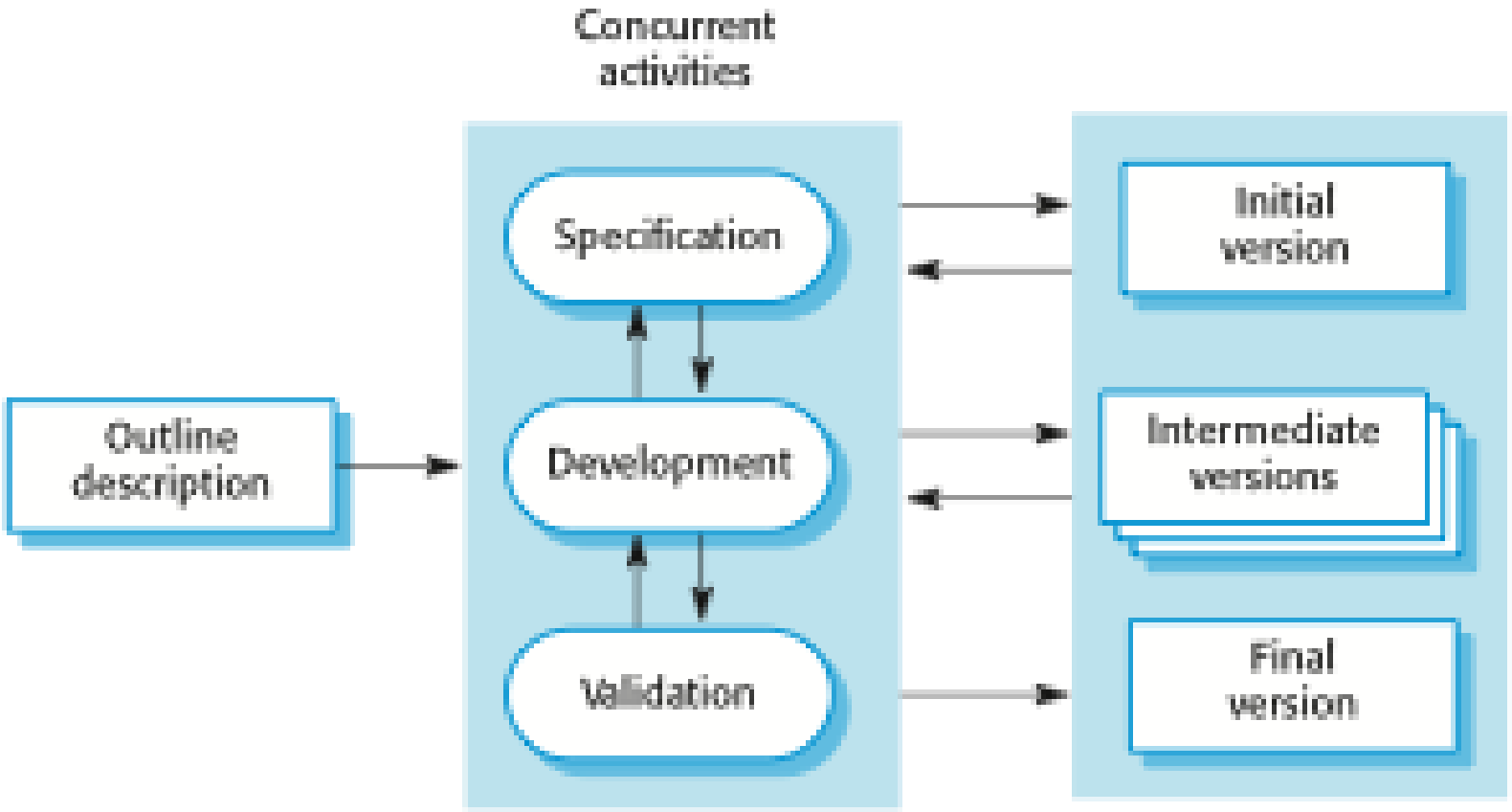


- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a 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.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Incremental development



Incremental development benefits

- The cost of accommodating changing customer requirements is reduced.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

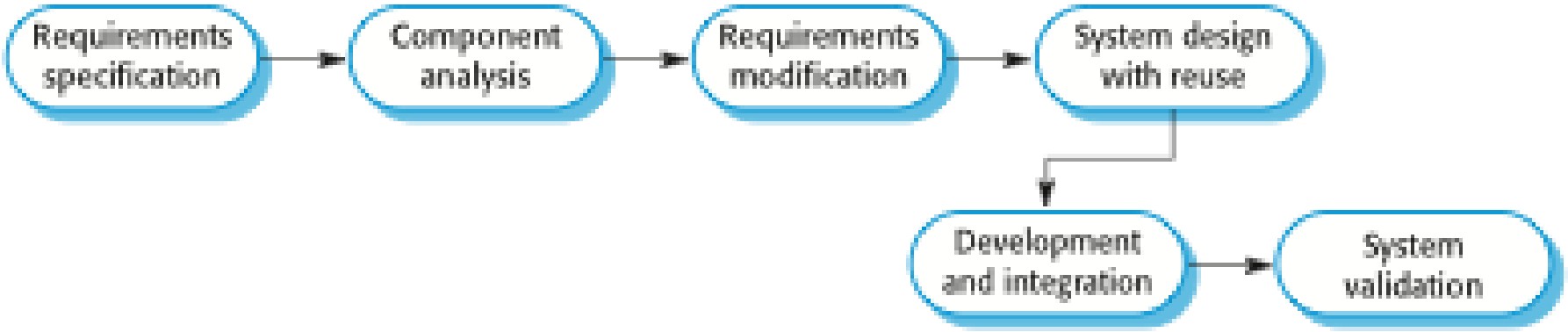
Incremental development problems

- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

Reuse-oriented 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.

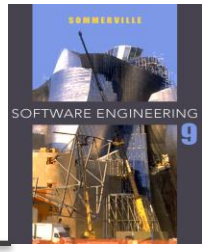
Reuse-oriented software engineering

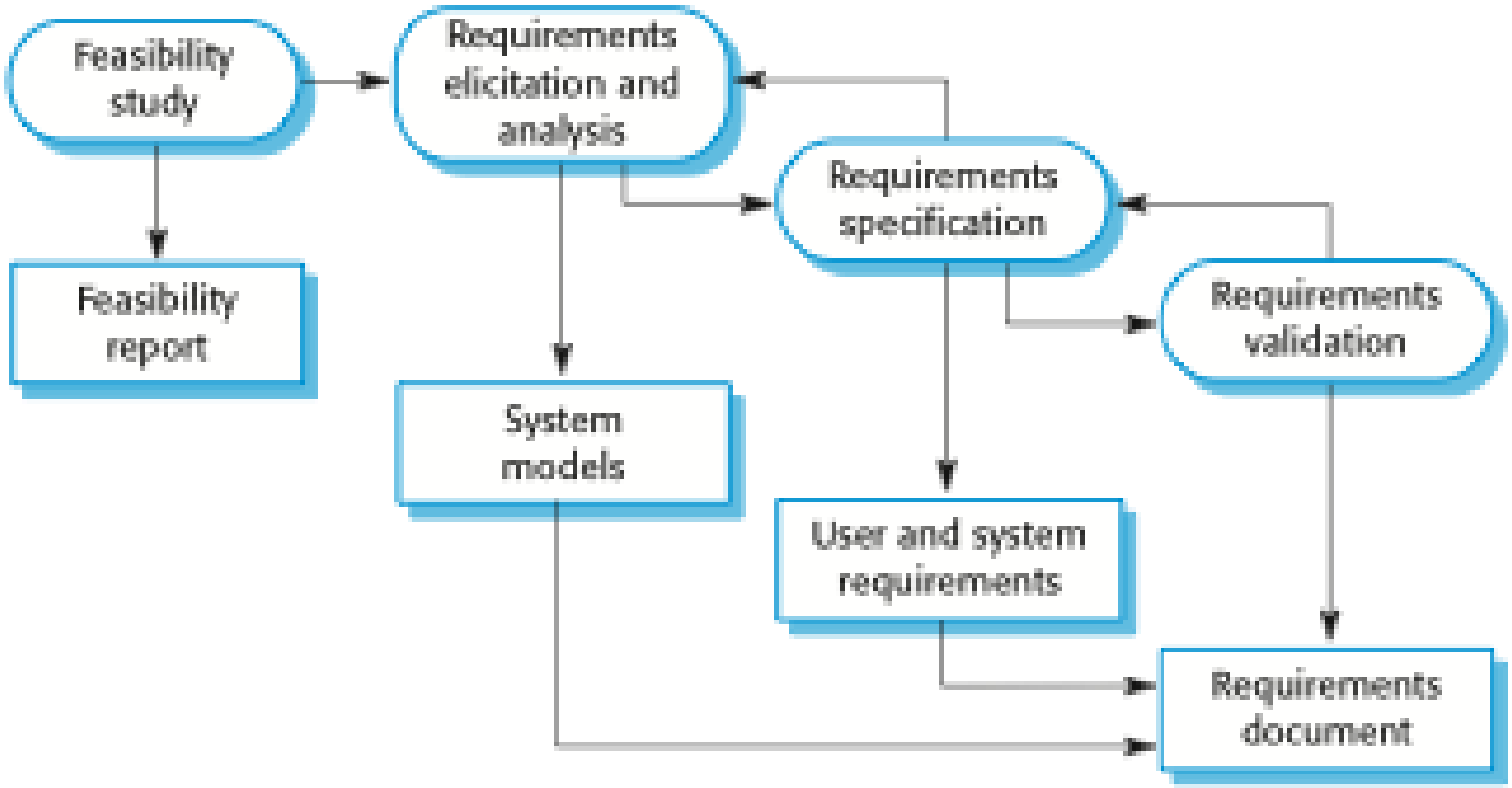


Types of software component

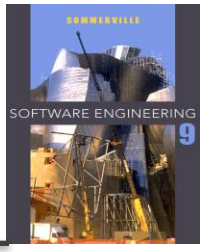
- Web services that are developed according to service standards and which are available for remote invocation.
- Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Stand-alone software systems (COTS) that are configured for use in a particular environment.

The requirements engineering process



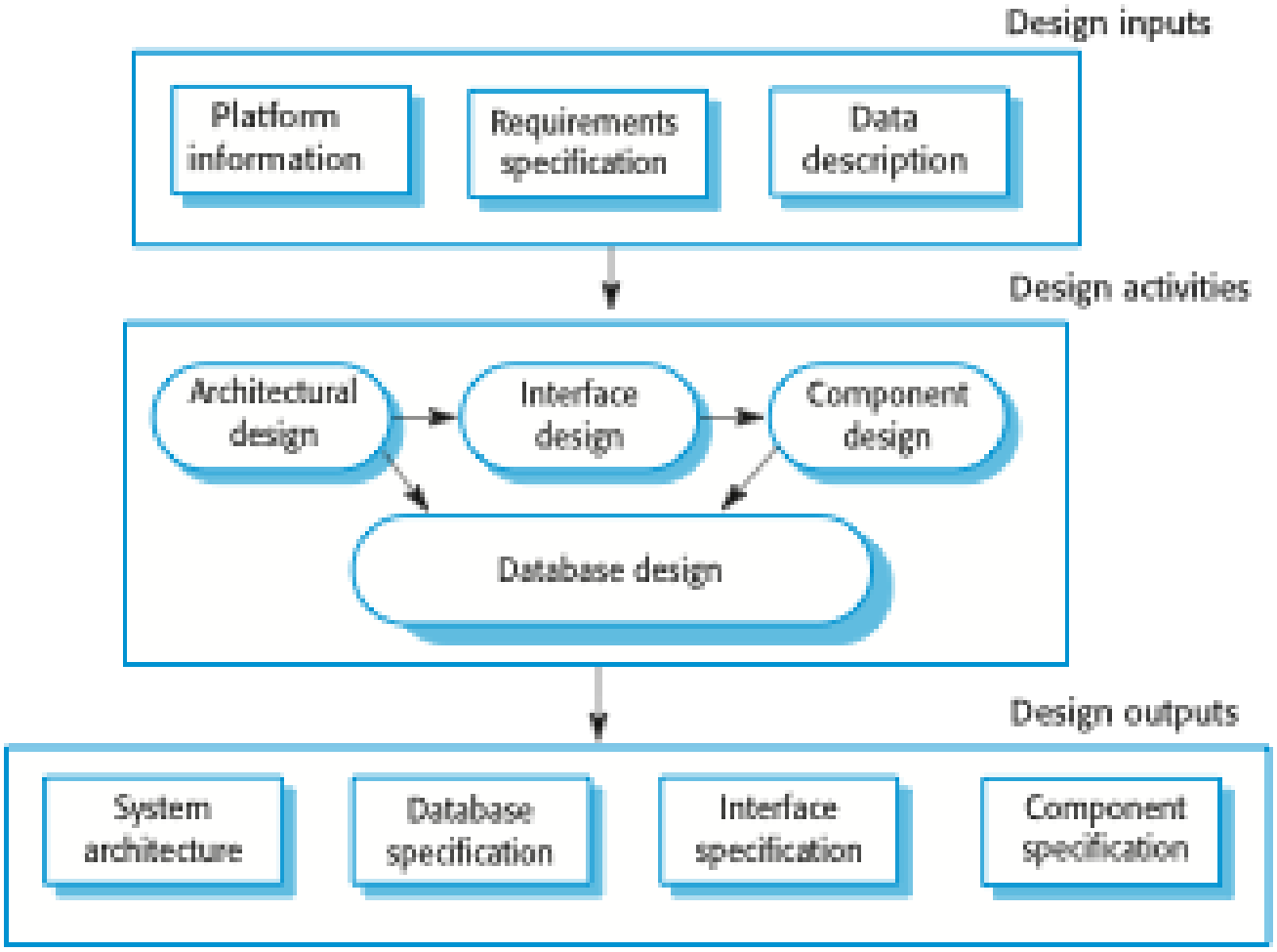


Software design and implementation

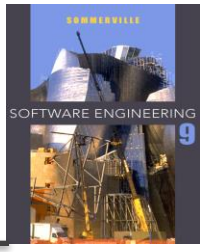


- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realises the specification;
- Implementation
  - Translate this structure into an executable program;
- The activities of design and implementation are closely related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

A general model of the design process



Design activities



- *Architectural design,* where you identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.
- *Interface design,* where you define the interfaces between system components.
- *Component design,* where you take each system component and design how it will operate.
- *Database design,* where you design the system data structures and how these are to be represented in a database.

Software validation

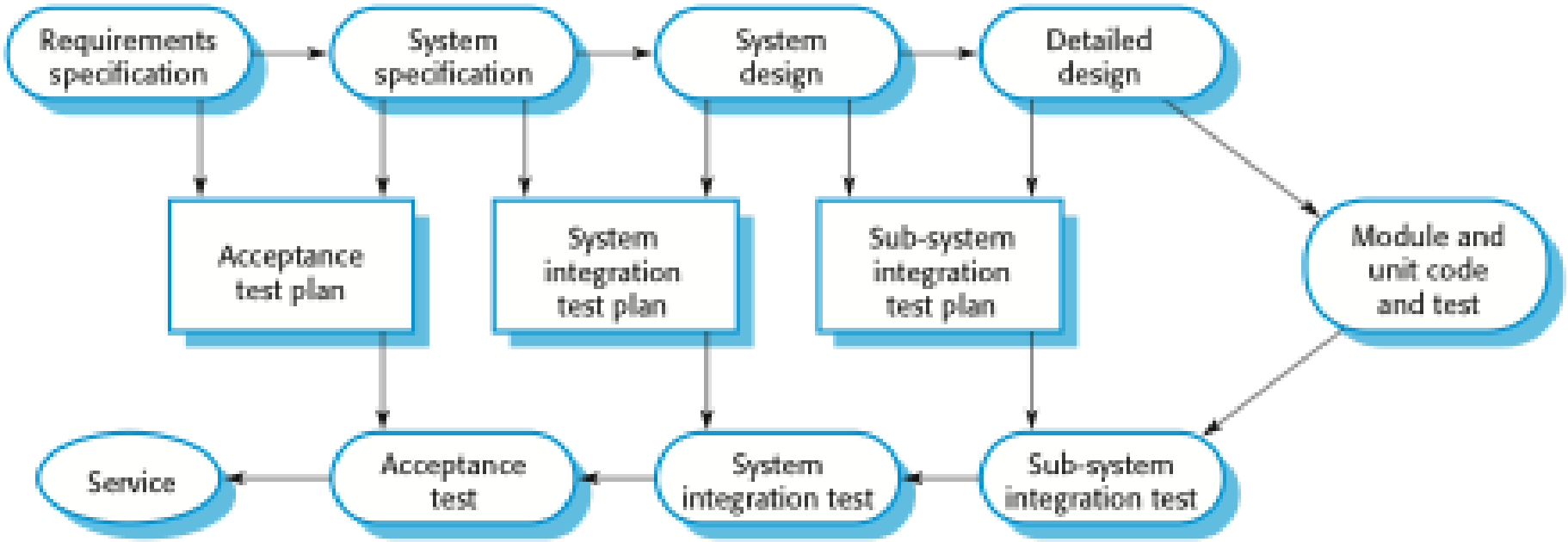
- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- Testing is the most commonly used V & V activity.

Stages of testing

Testing stages

- ✧ Development or component testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- ✧ System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- ✧ Acceptance testing
  - Testing with customer data to check that the system meets the customer's needs.

Testing phases in a plan-driven software process



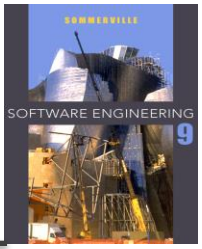
Coping with change

- Change is inevitable in all large software projects.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes
- Change leads to rework so the costs of change include both rework (e.g. re-analysing requirements) as well as the costs of implementing new functionality

Reducing the costs of rework

- Change avoidance, where the software process includes activities that can anticipate possible changes before significant rework is required.
  - For example, a prototype system may be developed to show some key features of the system to customers.
- Change tolerance, where the process is designed so that changes can be accommodated at relatively low cost.
  - This normally involves some form of incremental development. Proposed changes may be implemented in increments that have not yet been developed. If this is impossible, then only a single increment (a small part of the system) may have be altered to incorporate the change.

Software prototyping

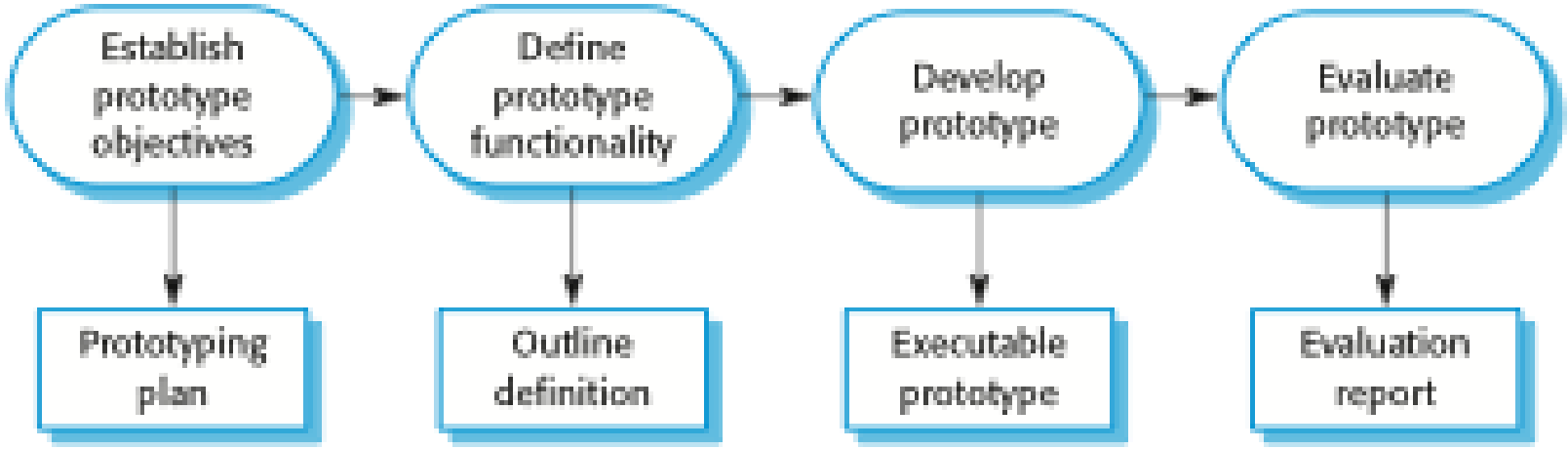


- A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- A prototype can be used in:
  - The requirements engineering process to help with requirements elicitation and validation;
  - In design processes to explore options and develop a UI design;
  - In the testing process to run back-to-back tests.

Benefits of prototyping

- ✧ Improved system usability.
- ✧ A closer match to users' real needs.
- ✧ Improved design quality.
- ✧ Improved maintainability.
- ✧ Reduced development effort.

The process of prototype development



Prototype development

- ✧ May be based on rapid prototyping languages or tools
- ✧ May involve leaving out functionality
  - Prototype should focus on areas of the product that are not well-understood;
  - Error checking and recovery may not be included in the prototype;
  - Focus on functional rather than non-functional requirements such as reliability and security

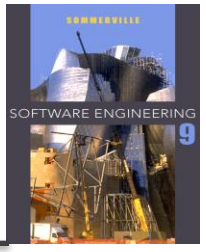
Throw-away prototypes

- Prototypes should be discarded after development as they are not a good basis for a production system:
  - It may be impossible to tune the system to meet non-functional requirements;
  - Prototypes are normally undocumented;
  - The prototype structure is usually degraded through rapid change;
  - The prototype probably will not meet normal organisational quality standards.

Incremental delivery

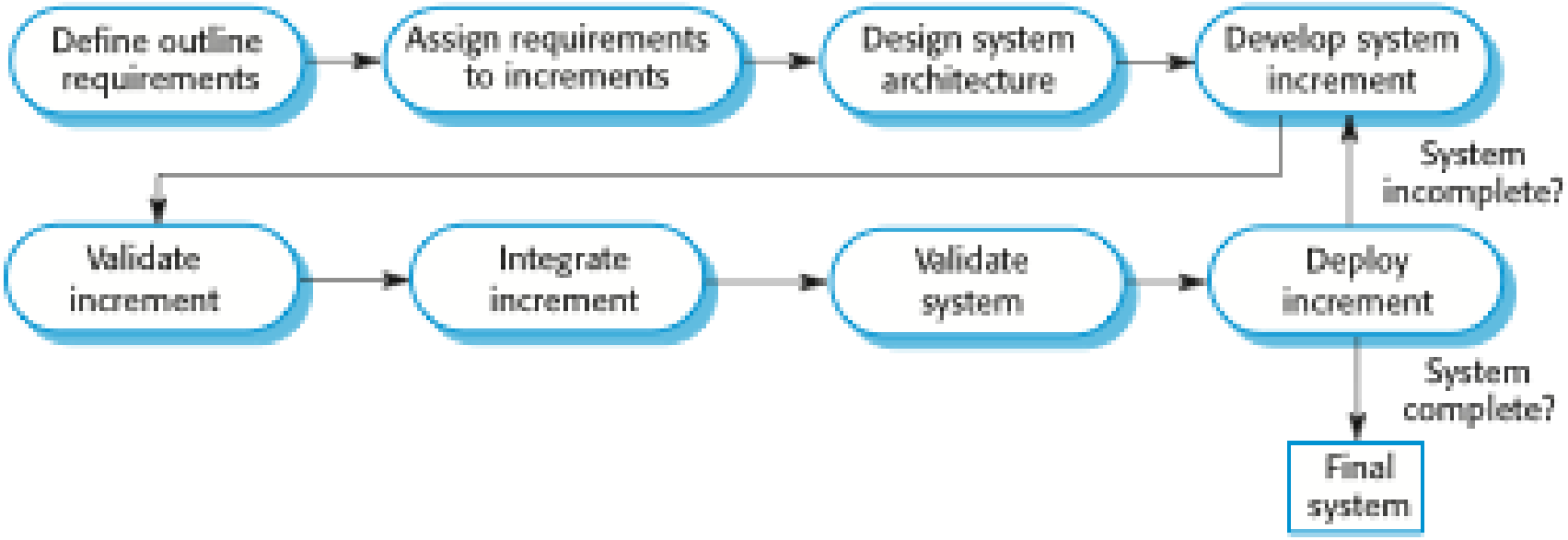
- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
- User requirements are prioritised and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

Incremental development and delivery



- Incremental development
  - Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
  - Normal approach used in agile methods;
  - Evaluation done by user/customer proxy.
- Incremental delivery
  - Deploy an increment for use by end-users;
  - More realistic evaluation about practical use of software;
  - Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

Incremental delivery



Incremental delivery advantages

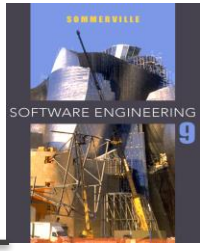
- Customer value can be delivered with each increment so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

Incremental delivery problems

- Most systems require a set of basic facilities that are used by different parts of the system.
  - As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
- The essence of iterative processes is that the specification is developed in conjunction with the software.
  - However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

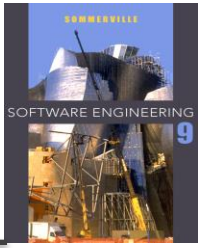
Process Improvement

Process Improvement



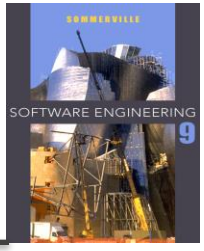
- Understanding, Modelling and Improving the Software Process

Objectives



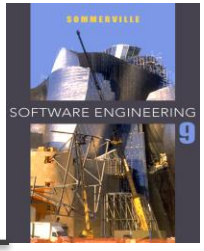
- To explain the principles of software process improvement
- To explain how software process factors influence software quality and productivity
- To introduce the SEI Capability Maturity Model and to explain why it is influential. To discuss the applicability of that model
- To explain why CMM-based improvement is not universally applicable

Topics covered



- ✧ Process and product quality
- ✧ Process analysis and modelling
- ✧ Process measurement
- ✧ The SEI process maturity model
- ✧ Process classification

Process improvement

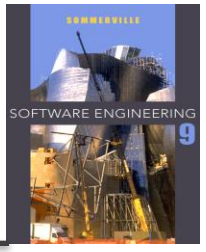


- Understanding existing processes
- Introducing process changes to achieve organisational objectives which are usually focused on quality improvement, cost reduction and schedule acceleration
- Most process improvement work so far has focused on defect reduction. This reflects the increasing attention paid by industry to quality
- However, other process attributes can be the focus of improvement

Process attributes

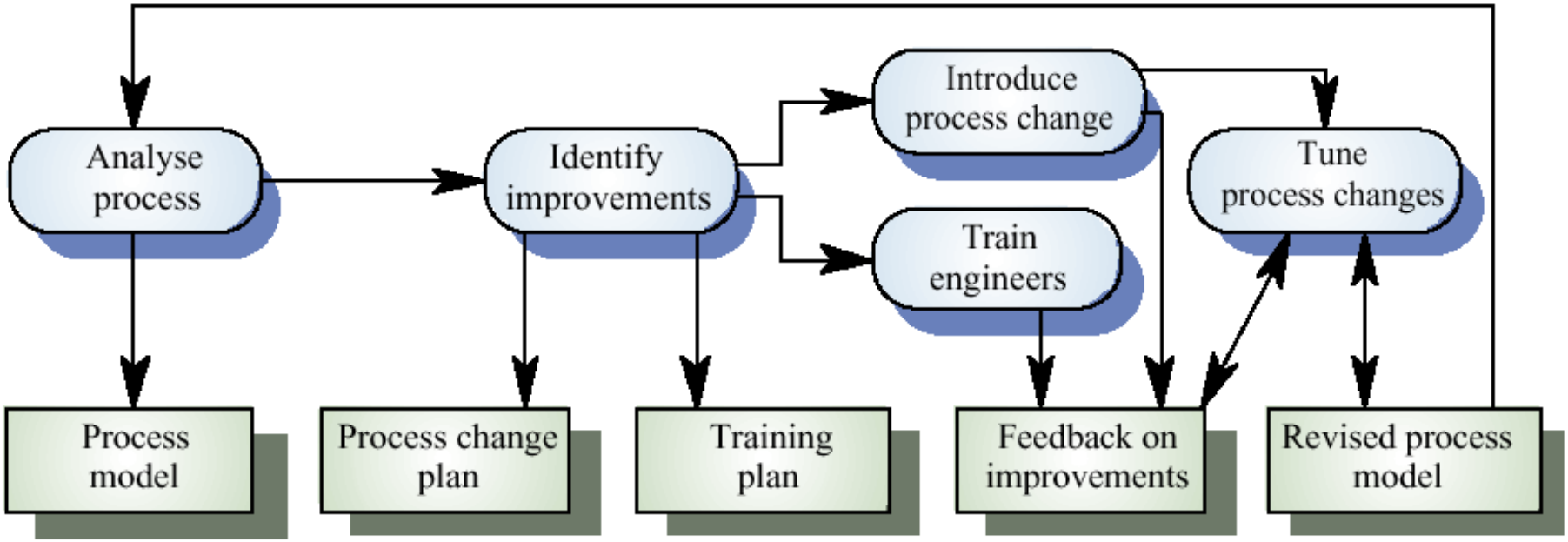
Process characteristic	Description
Understandability	To what extent is the process explicitly defined and how easy is it to understand the process definition?
Visibility	Do the process activities culminate in clear results so that the progress of the process is externally visible?
Supportability	To what extent can the process activities be supported by CASE tools?
Acceptability	Is the defined process acceptable to and usable by the engineers responsible for producing the software product?
Reliability	Is the process designed in such a way that process errors are avoided or trapped before they result in product errors?
Robustness	Can the process continue in spite of unexpected problems?
Maintainability	Can the process evolve to reflect changing organisational requirements or identified process improvements?
Rapidity	How fast can the process of delivering a system from a given specification be completed?

Process improvement stages

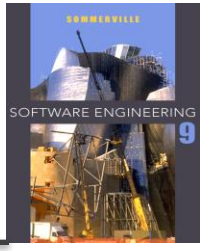


- Process analysis
  - Model and analyse (quantitatively if possible) existing processes
- Improvement identification
  - Identify quality, cost or schedule bottlenecks
- Process change introduction
  - Modify the process to remove identified bottlenecks
- Process change training
  - Train staff involved in new process proposals
- Change tuning
  - Evolve and improve process improvements

The process improvement proc

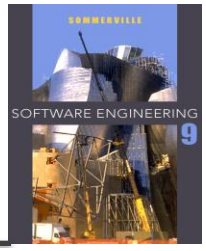


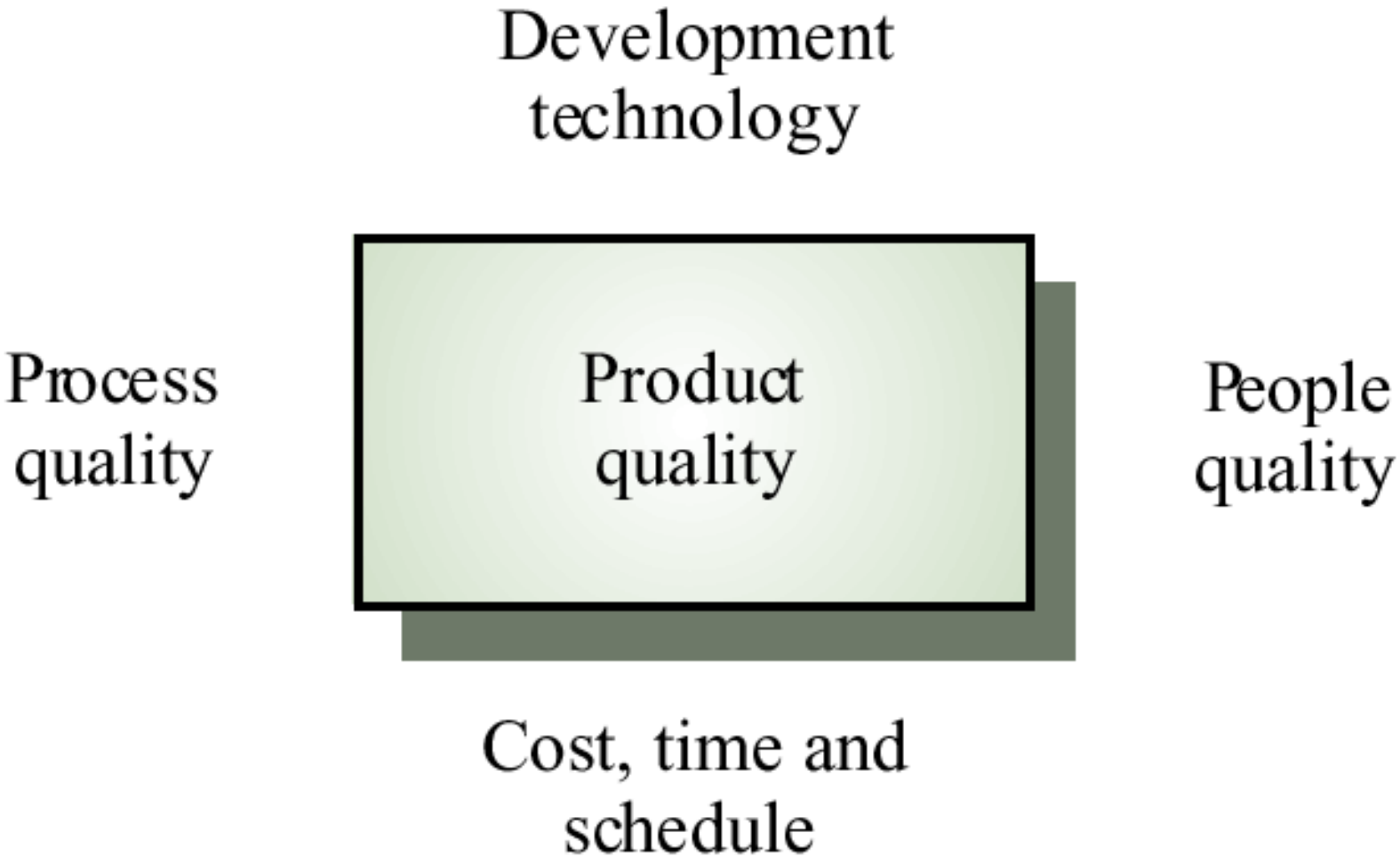
Process and product quality



- Process quality and product quality are closely related
- A good process is usually required to produce a good product
- For manufactured goods, process is the principal quality determinant
- For design-based activity, other factors are also involved especially the capabilities of the designers

Principal product quality factors

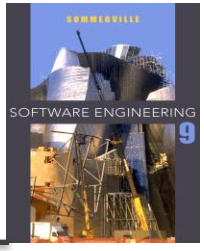




Quality factors

- For large projects with ‘average’ capabilities, the development process determines product quality
- For small projects, the capabilities of the developers is the main determinant
- The development technology is particularly significant for small projects
- In all cases, if an unrealistic schedule is imposed then product quality will suffer

Process analysis and modelling



- Process analysis
  - The study of existing processes to understand the relationships between parts of the process and to compare them with other processes
- Process modelling
  - The documentation of a process which records the tasks, the roles and the entities used
  - Process models may be presented from different perspectives

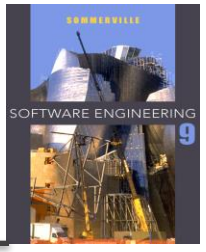
Process analysis and modelling

- Study an existing process to understand its activities
- Produce an abstract model of the process. You should normally represent this graphically. Several different views (e.g. activities, deliverables, etc.) may be required
- Analyse the model to discover process problems. Involves discussing activities with stakeholders

Process analysis techniques

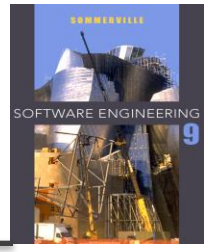
- Published process models and process standards
  - It is always best to start process analysis with an existing model. People then may extend and change this.
- Questionnaires and interviews
  - Must be carefully designed. Participants may tell you what they think you want to hear
- Ethnographic analysis
  - Involves assimilating process knowledge by observation

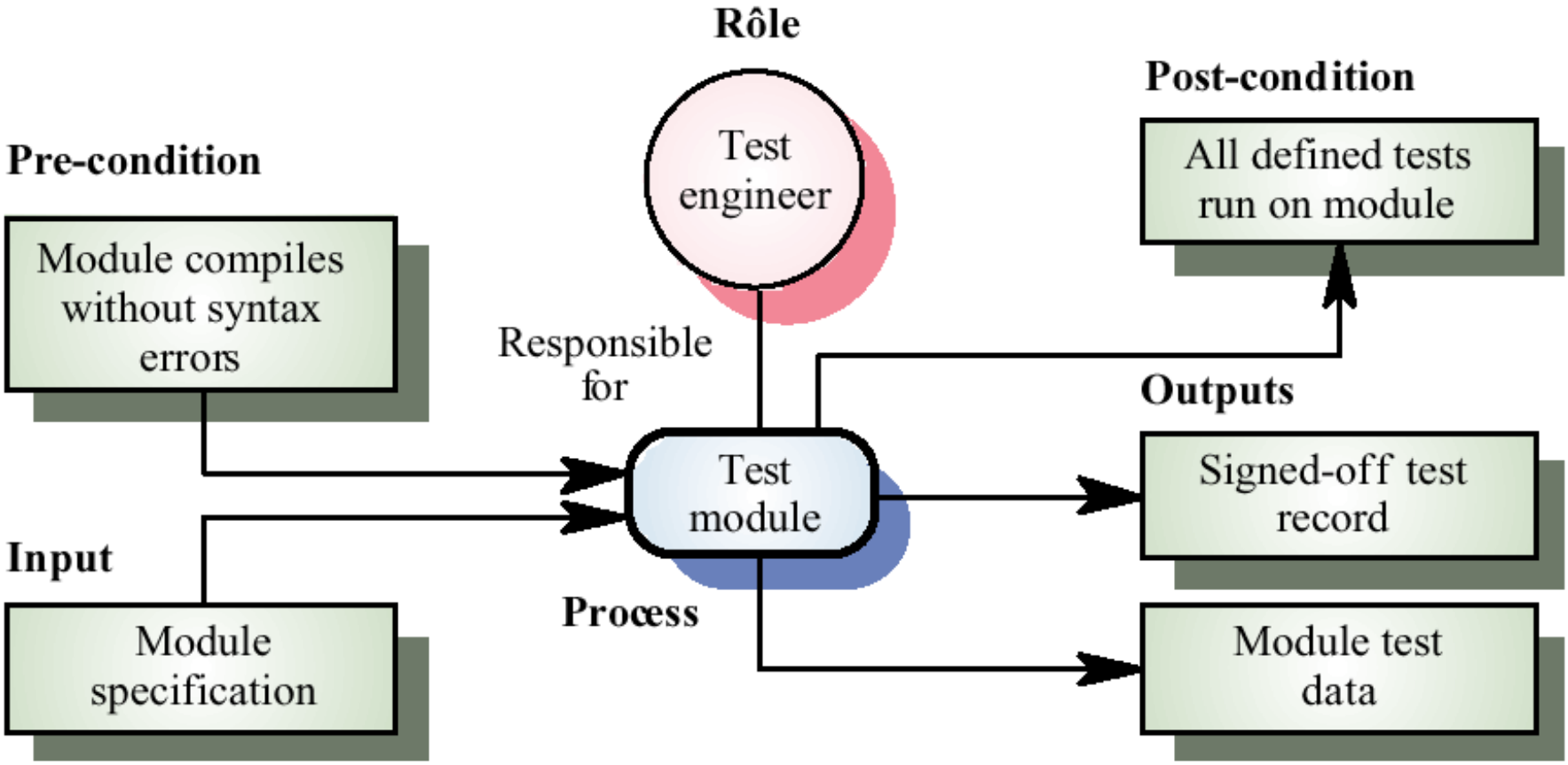
Process model element	Description
Activity (represented by a round-edged rectangle with no drop shadow)	An activity has a clearly defined objective, entry and exit conditions. Examples of activities are preparing a set of test data to test a module, coding a function or a module, proof-reading a document, etc. Generally, an activity is atomic i.e. it is the responsibility of one person or group. It is not decomposed into sub-activities.
Process (represented by a round-edged rectangle with drop shadow)	A process is a set of activities which have some coherence and whose objective is generally agreed within an organisation. Examples of processes are requirements analysis, architectural design, test planning, etc.
Deliverable (represented by a rectangle with drop shadow)	A deliverable is a tangible output of an activity which is predicted in a project plan.
Condition (represented by a parallelogram)	A condition is either a pre-condition which must hold before a process or activity can start or a post-condition which holds after a process or activity has finished.
Role (represented by a circle with drop shadow)	A role is a bounded area of responsibility. Examples of roles might be configuration manager, test engineer, software designer, etc. One person may have several different roles and a single role may be associated with several different people.
Exception (not shown in examples here but may be represented as a double edged box)	An exception is a description of how to modify the process if some anticipated or unanticipated event occurs. Exceptions are often undefined and it is left to the ingenuity of the project managers and engineers to handle the exception.
Communication (represented by an arrow)	An interchange of information between people or between people and supporting computer systems. Communications may be informal or formal. Formal communications might be the approval of a deliverable by a project manager; informal communications might be the interchange of electronic mail to resolve ambiguities in a document.



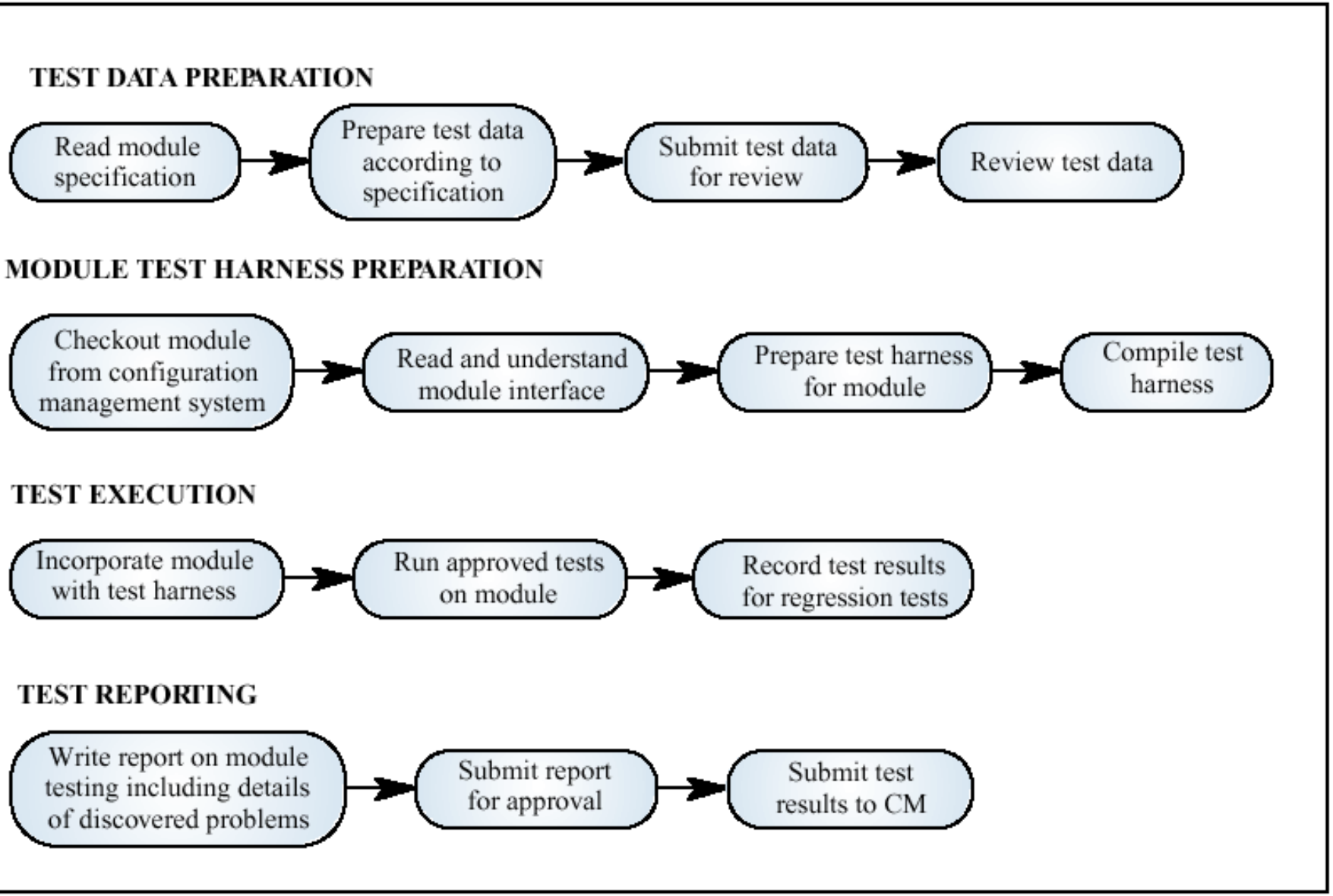
Elements of a process model

The module testing activity





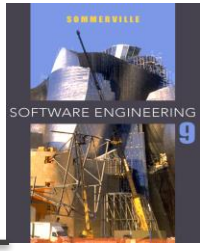
Activities in module testing



Process exceptions

- Software processes are complex and process models cannot effectively represent how to handle exceptions
  - Several key people becoming ill just before a critical review
  - A complete failure of a communication processor so that no e-mail is available for several days
  - Organisational reorganisation
  - A need to respond to an unanticipated request for new proposals
- Under these circumstances, the model is suspended and managers use their initiative to deal with the exception

Process measurement



- Wherever possible, quantitative process data should be collected
  - However, where organisations do not have clearly defined process standards this is very difficult as you don't know what to measure. A process may have to be defined before any measurement is possible
- Process measurements should be used to assess process improvements
  - But this does not mean that measurements should drive the improvements. The improvement driver should be the organizational objectives

Classes of process measurement

- Time taken for process activities to be completed
  - E.g. Calendar time or effort to complete an activity or process
- Resources required for processes or activities
  - E.g. Total effort in person-days
- Number of occurrences of a particular event
  - E.g. Number of defects discovered

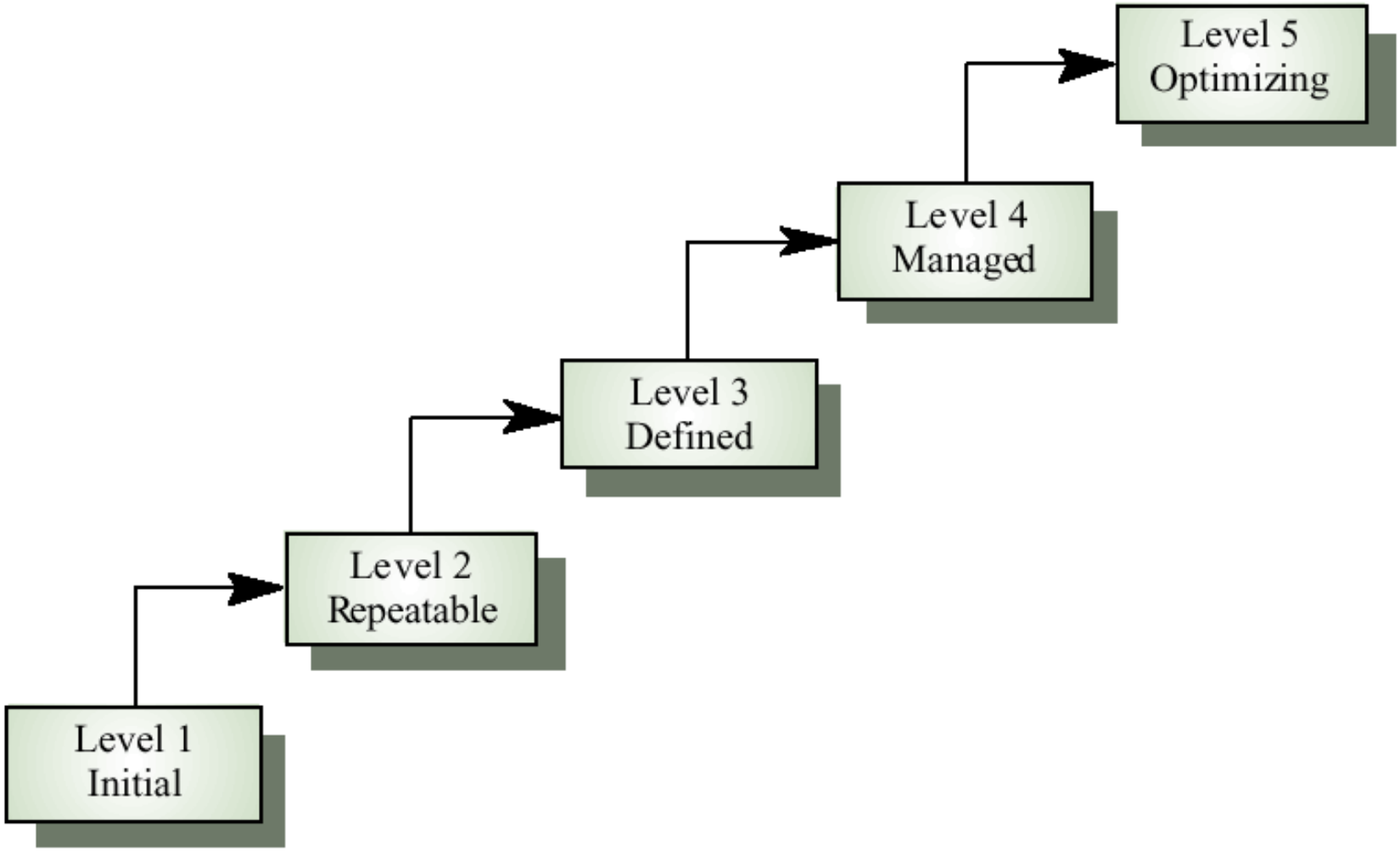
Goal-Question-Metric Paradigm

- Goals
  - What is the organisation trying to achieve? The objective of process improvement is to satisfy these goals
- Questions
  - Questions about areas of uncertainty related to the goals. You need process knowledge to derive these
- Metrics
  - Measurements to be collected to answer the questions

The Software Engineering Institute

- US Defense Dept. funded institute associated with Carnegie Mellon
- Mission is to promote software technology transfer particularly to defense contractors
- Maturity model proposed in mid-1980s, refined in early 1990s.
- Work has been very influential in process improvement

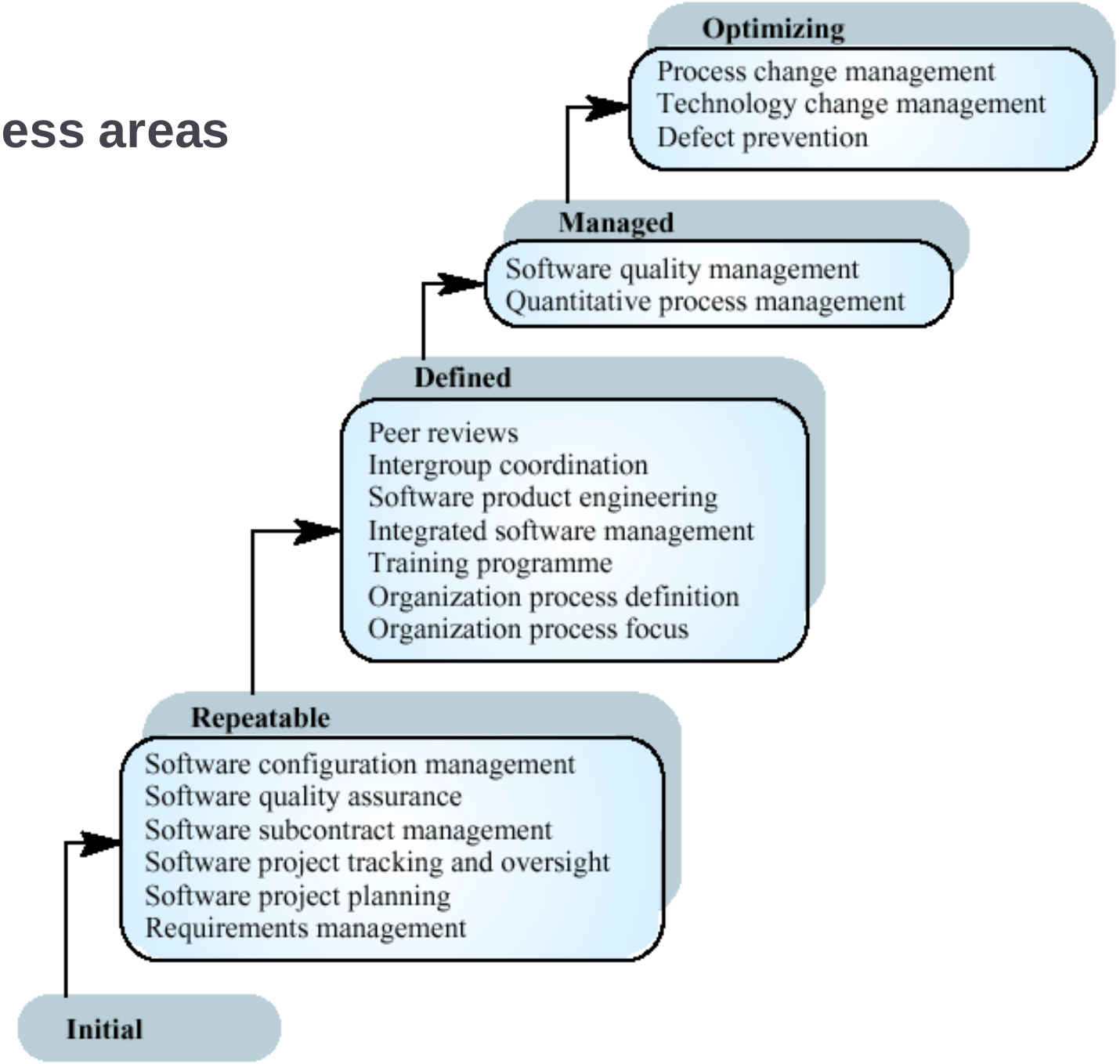
The SEI process maturity model



Maturity model levels

- ✧ Initial
  - Essentially uncontrolled
- ✧ Repeatable
  - Product management procedures defined and used
- ✧ Defined
  - Process management procedures and strategies defined and used
- ✧ Managed
  - Quality management strategies defined and used
- ✧ Optimising
  - Process improvement strategies defined and used

Key process areas



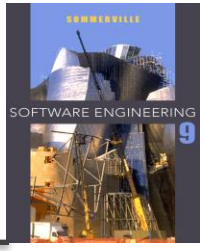
SEI model problems

- It focuses on project management rather than product development.
- It ignores the use of technologies such as rapid prototyping.
- It does not incorporate risk analysis as a key process area
- It does not define its domain of applicability

The CMM and ISO 9000

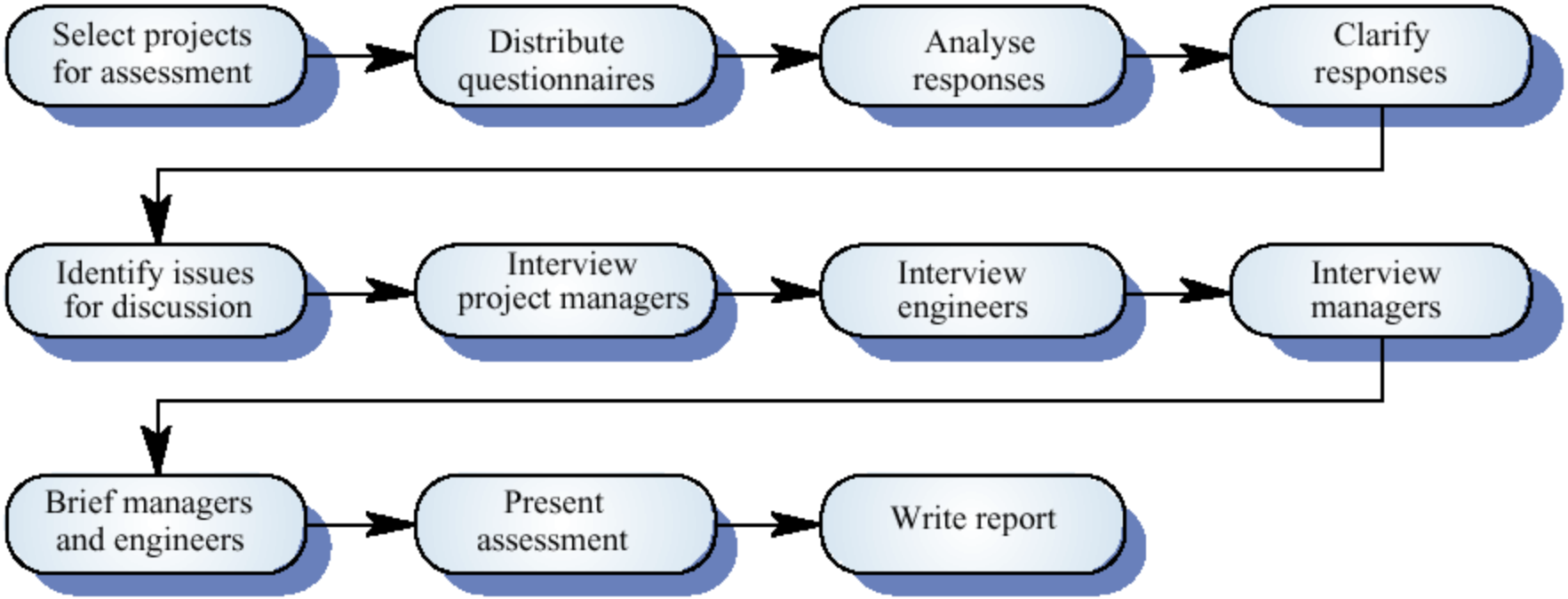
- There is a clear correlation between the key processes in the CMM and the quality management processes in ISO 9000
- The CMM is more detailed and prescriptive and includes a framework for improvement
- Organisations rated as level 2 in the CMM are likely to be ISO 9000 compliant

Capability assessment



- An important role of the SEI is to use the CMM to assess the capabilities of contractors bidding for US government defence contracts
- The model is intended to represent organisational capability not the practices used in particular projects
- Within the same organisation, there are often wide variations in processes used
- Capability assessment is questionnaire-based

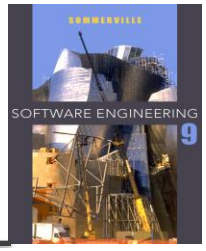
The capability assessment proce

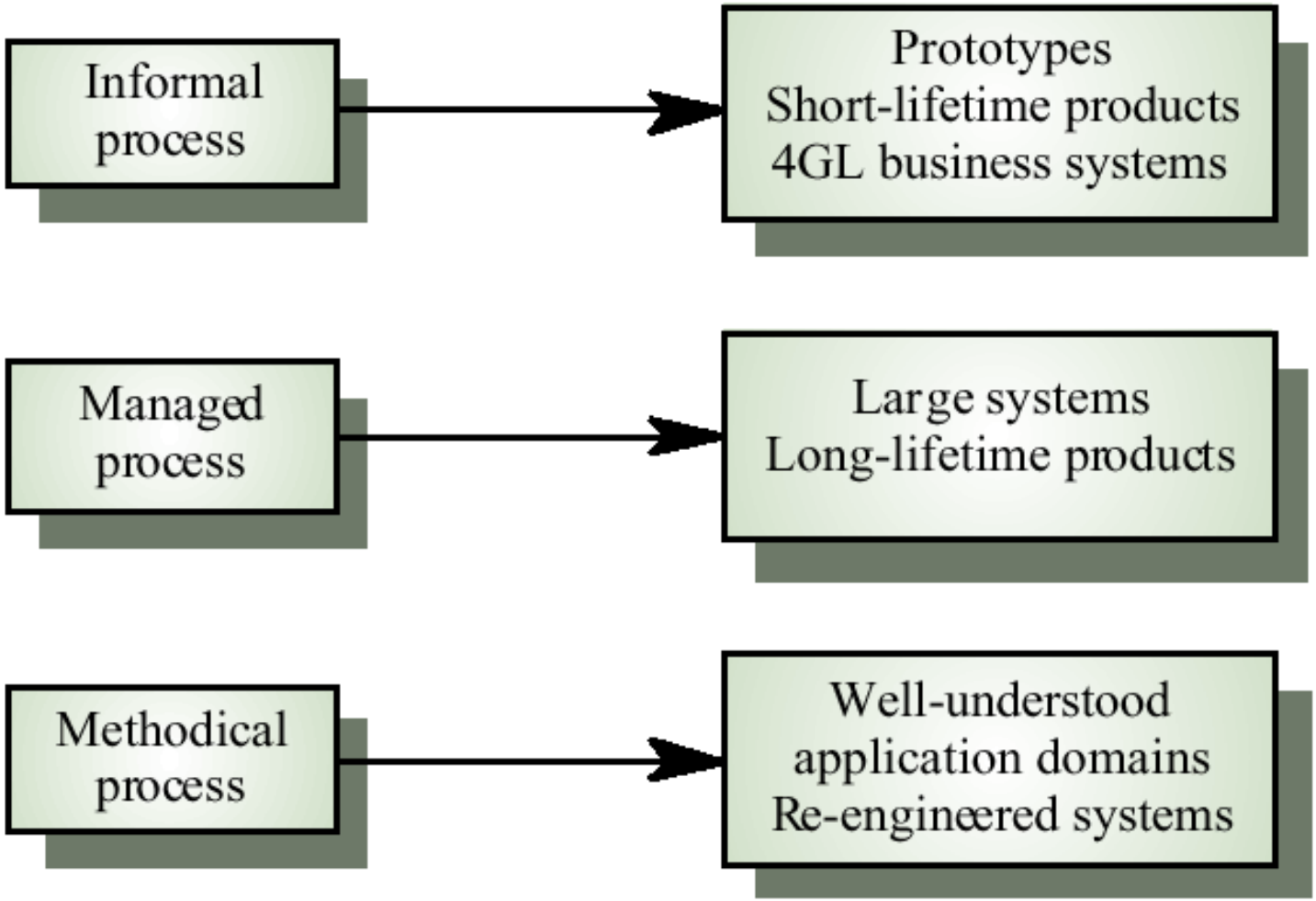


Process classification

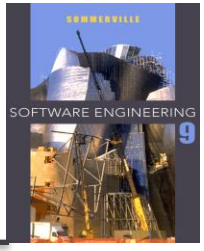
- ✧ Informal
  - No detailed process model. Development team chose their own way of working
- ✧ Managed
  - Defined process model which drives the development process
- ✧ Methodical
  - Processes supported by some development method such as HOOD
- ✧ Supported
  - Processes supported by automated CASE tools

Process applicability



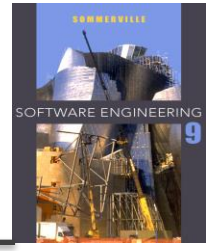


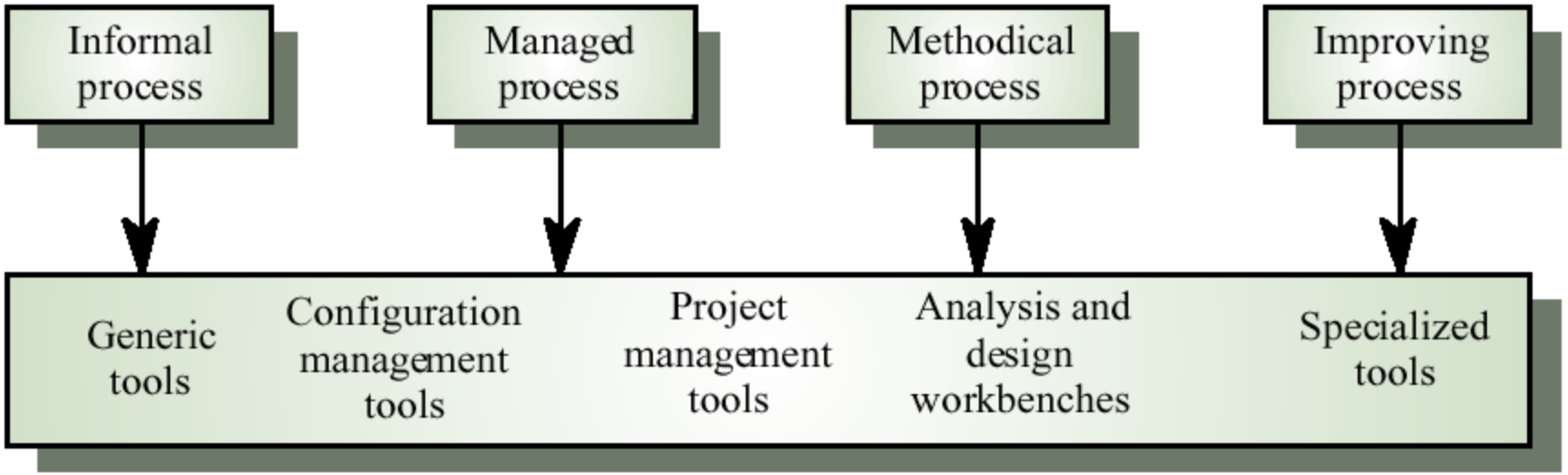
Process choice



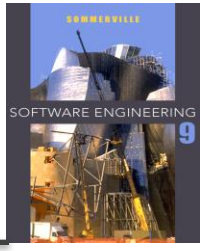
- Process used should depend on type of product which is being developed
  - For large systems, management is usually the principal problem so you need a strictly managed process. For smaller systems, more informality is possible.
- There is no uniformly applicable process which should be standardised within an organisation
  - High costs may be incurred if you force an inappropriate process on a development team

Process tool support



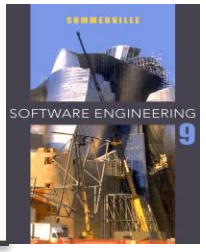


Key points



- Process improvement involves process analysis, standardisation, measurement and change
- Process models include descriptions of tasks, activities, roles, exceptions, communications, deliverables and other processes
- Measurement should be used to answer specific questions about the software process used
- The three types of process metrics which can be collected are time metrics, resource utilisation metrics and event metrics

Key points



- The Software Engineering Institute (SEI) model classifies software processes as initial, repeatable, defined, managed and optimising. It identifies key processes which should be used at each of these levels
- The SEI model is appropriate for large systems developed by large teams of engineers. It cannot be applied without modification in other situations
- Processes can be classified as informal, managed, methodical and improving. This classification can be used to identify process tool support