

Software Processes

SCSJ 2203 Software Engineering

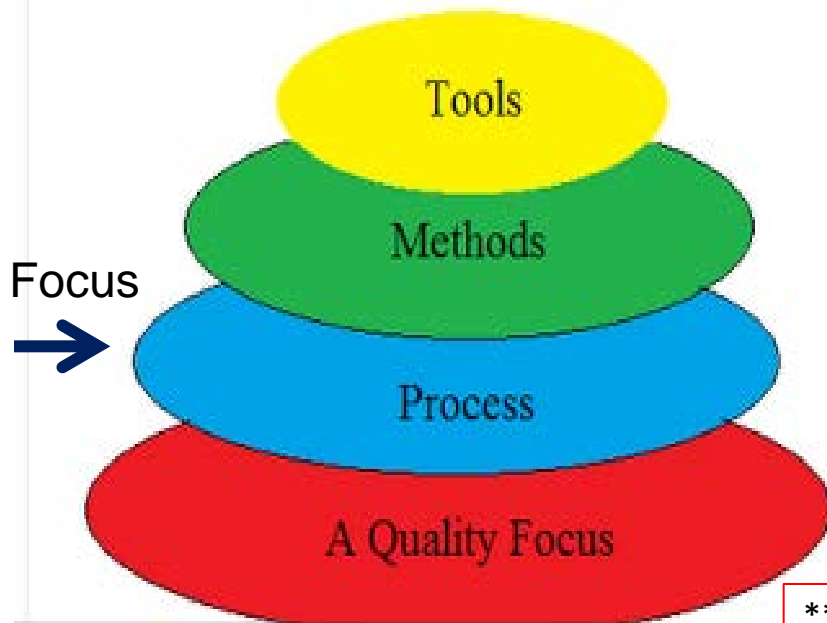
Topics covered

- Software process models
- Process activities
- Coping with change
- The Rational Unified Process
 - An example of a modern software process.

Recap:: What is Software Engineering?

www.utm.my

Software Engineering as Layered Technology



- **Quality Focus:** The bedrock that supports software engineering is quality focus where it ensures continuous process improvement culture.
- **Process:** Foundation for software engineering which enables rational and timely development of computer software
- **Methods:** provide technical how to's for building software. Involve different tasks including requirements analysis, design, program construction, testing and support. Methods also include modeling activities
- **Tools:** provide automated or semi-automated support for the process and methods

** The layered technology will be the focus in this lecture and throughout the semesters

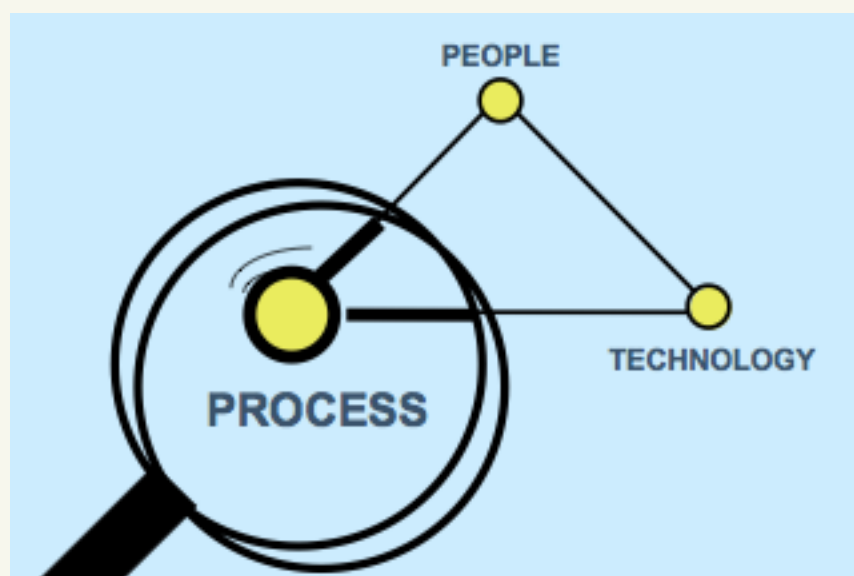
Source:

Agarwal, U. (2012). *Software Engineering*, Kataria and Sons.

<http://sesolution.blogspot.my/p/software-engineering-layered-technology.html>

The Role of Process

- Even the finest people can't perform at their best when the process is not understood or operating at its best.



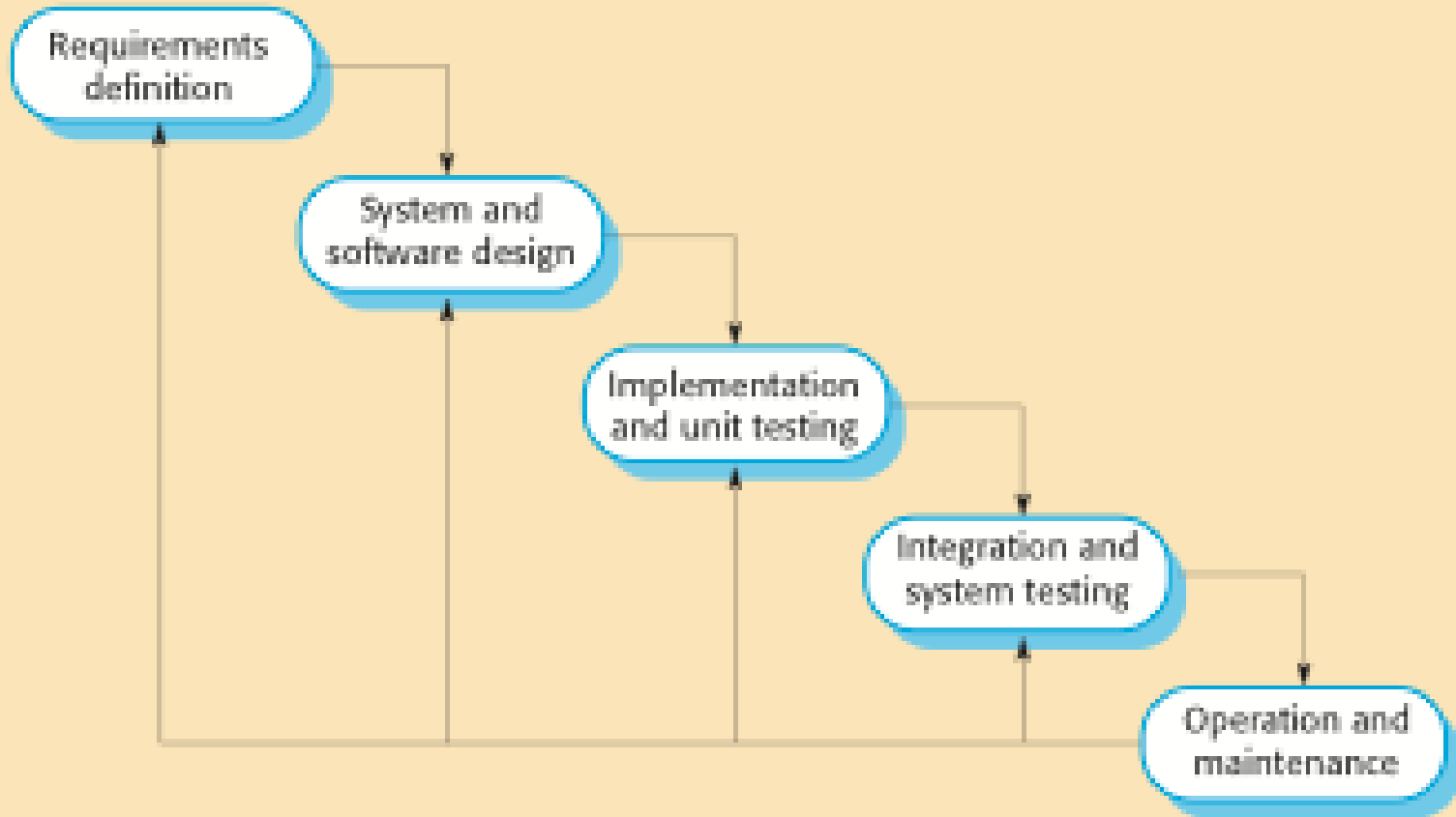
The software process

- 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 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



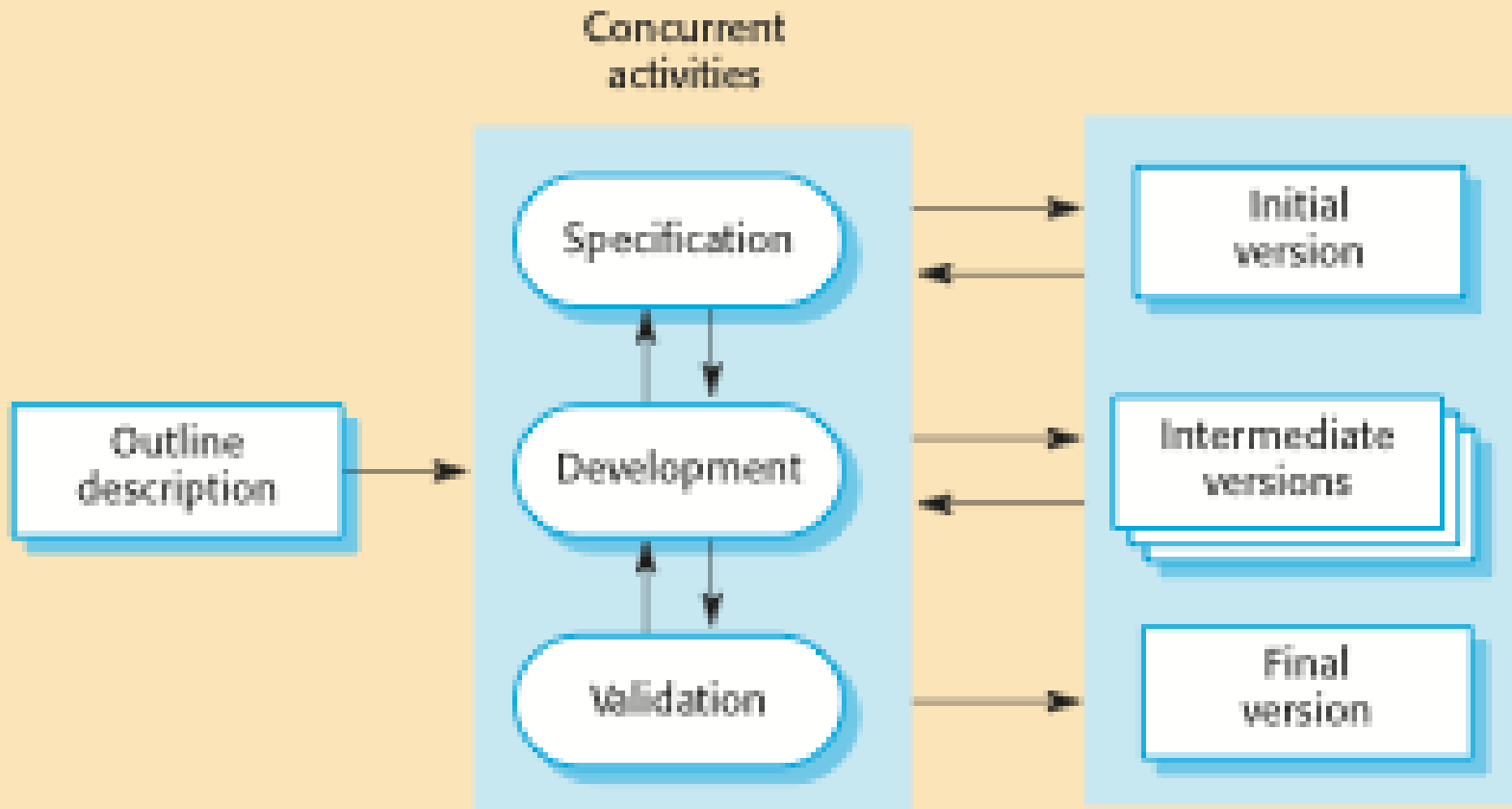
Waterfall model phases

- 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.

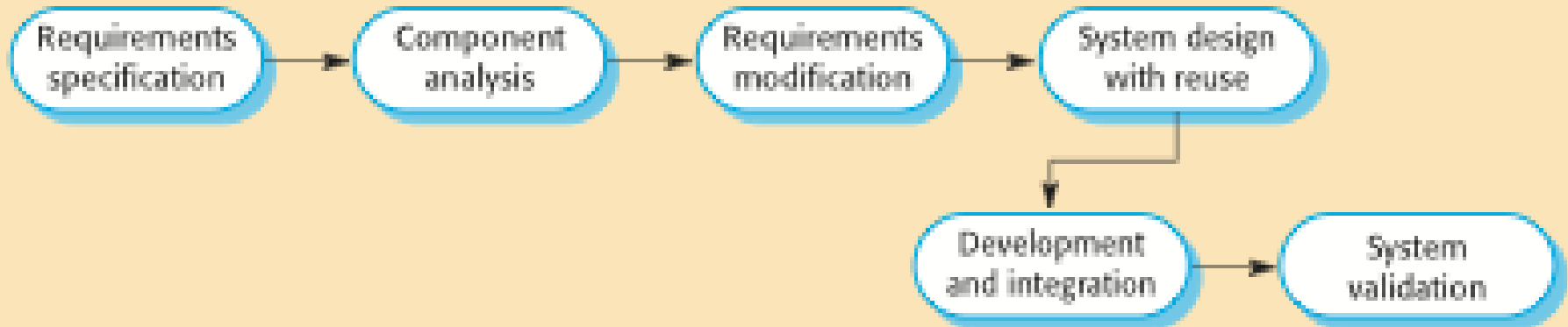
Incremental development problems

- 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 is now the standard approach for building many types of business system
 - Reuse covered in more depth in Chapter 16.

Reuse-oriented software engineering

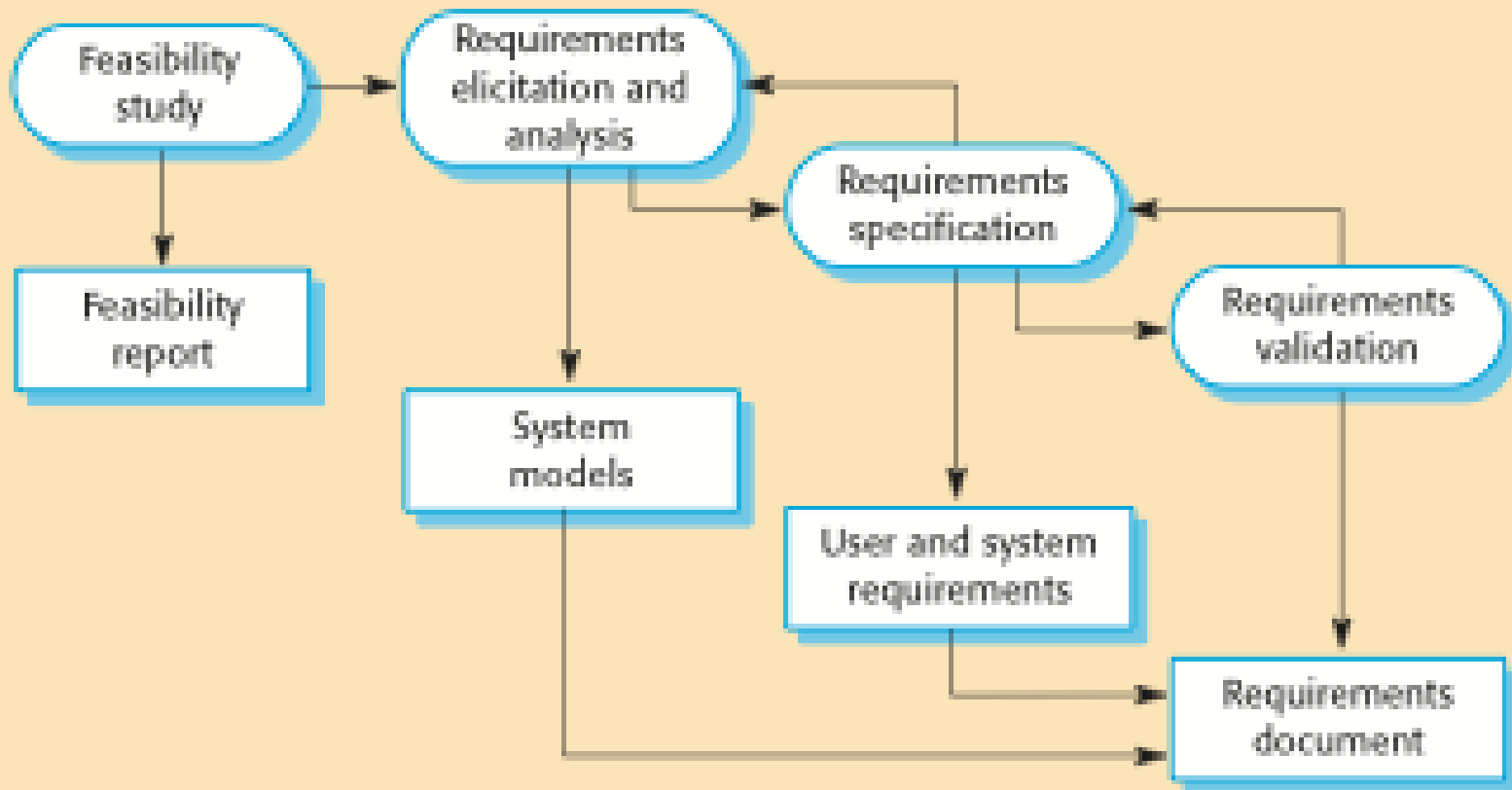


Types of software component

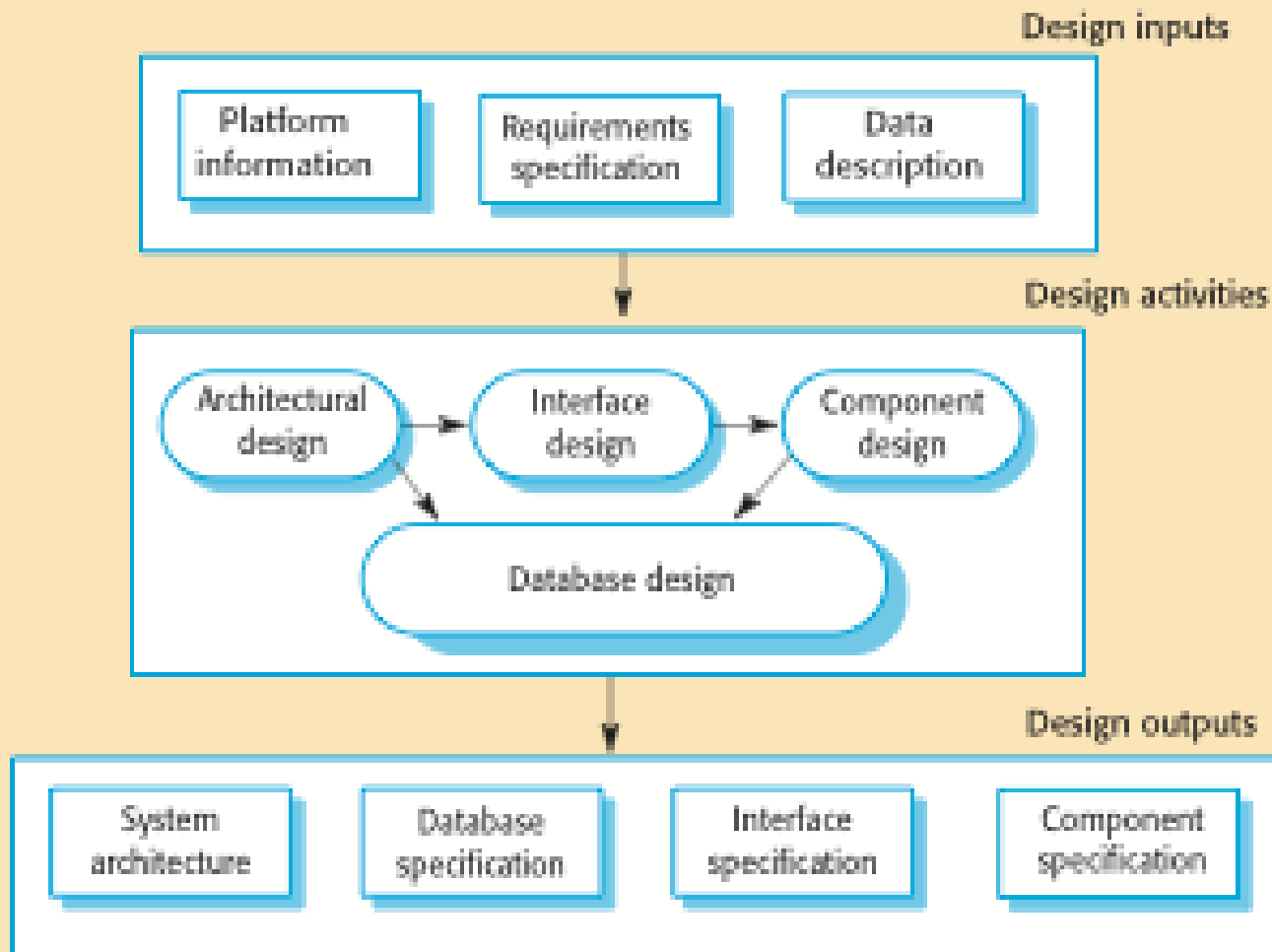
- Web services that are developed according to service standards and which are available for remote invocation.
- Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Stand-alone software systems (COTS) that are configured for use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

Process activities

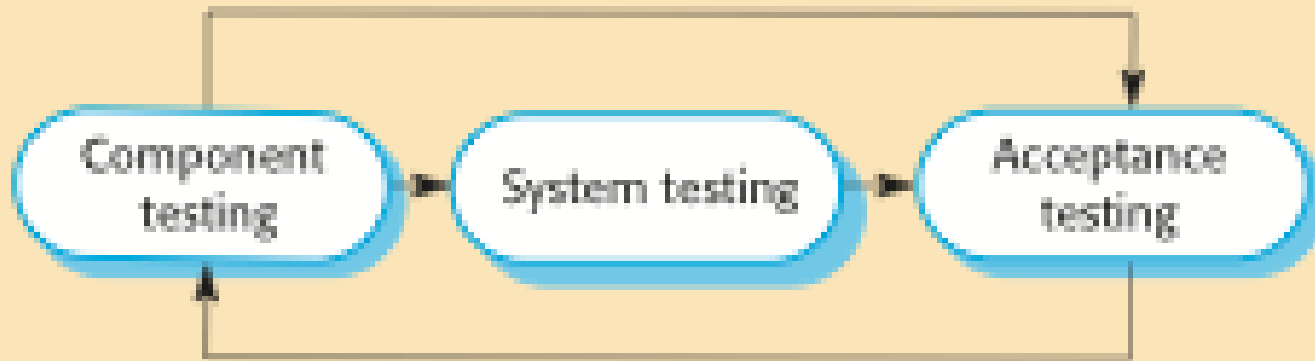
- Real software processes are inter-leaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.
- The four basic process activities of specification, development, validation and evolution are organized differently in different development processes. In the waterfall model, they are organized in sequence, whereas in incremental development they are inter-leaved.



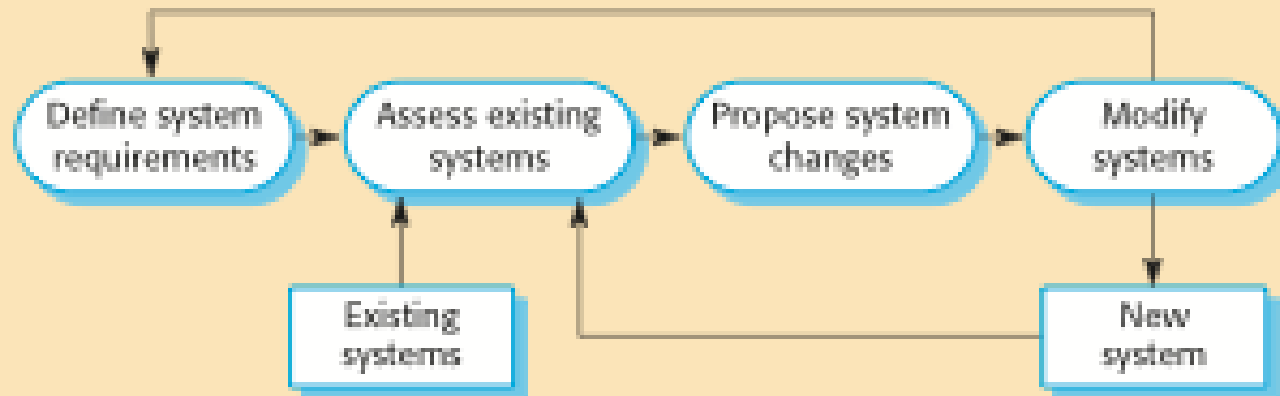
A general model of the design process



Stages of testing



System evolution



PART II

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.

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.

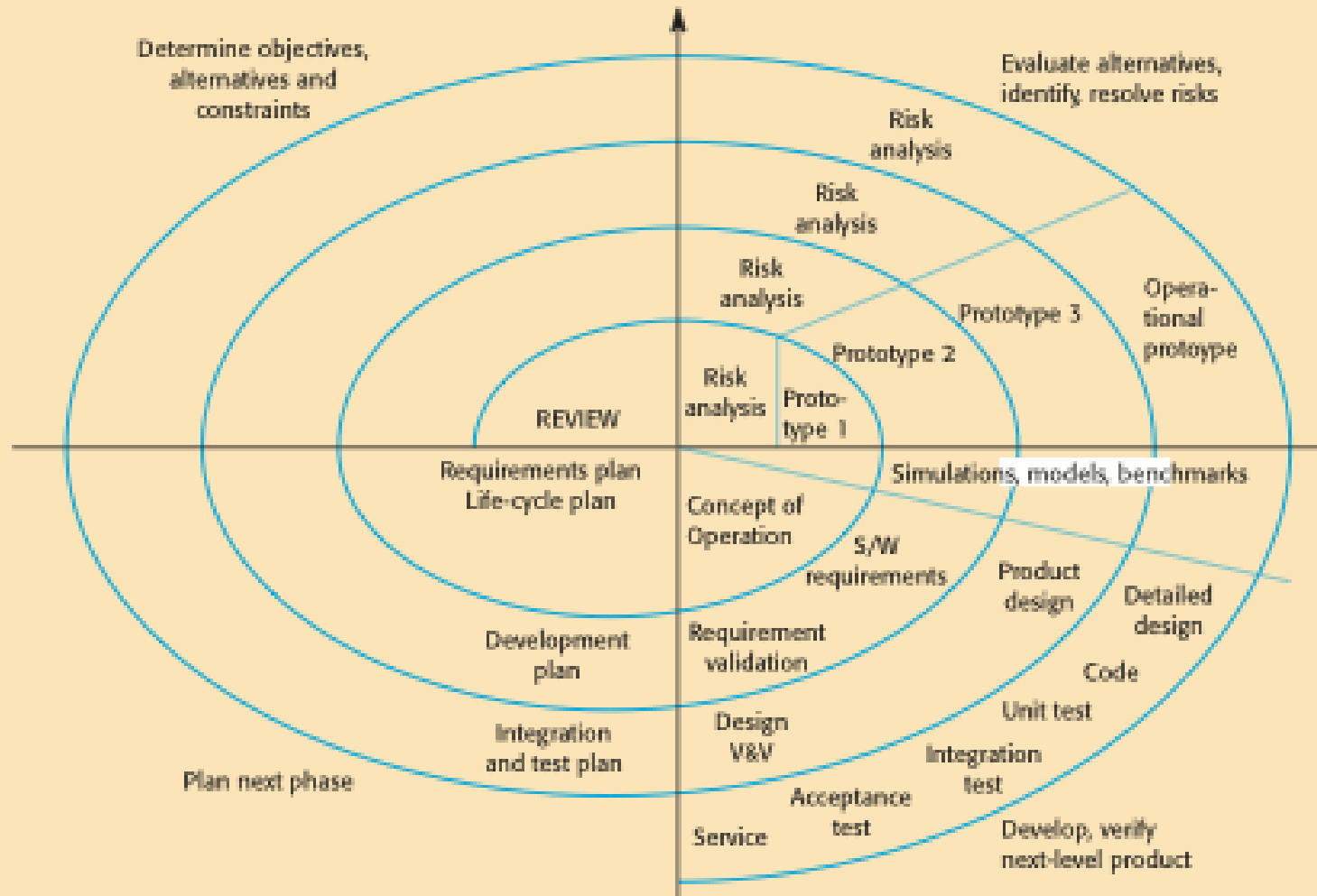
Incremental delivery advantages

- 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.

Boehm's spiral model

- 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.

Boehm's 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.

Spiral model usage

- Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.
- In practice, however, the model is rarely used as published for practical software development.

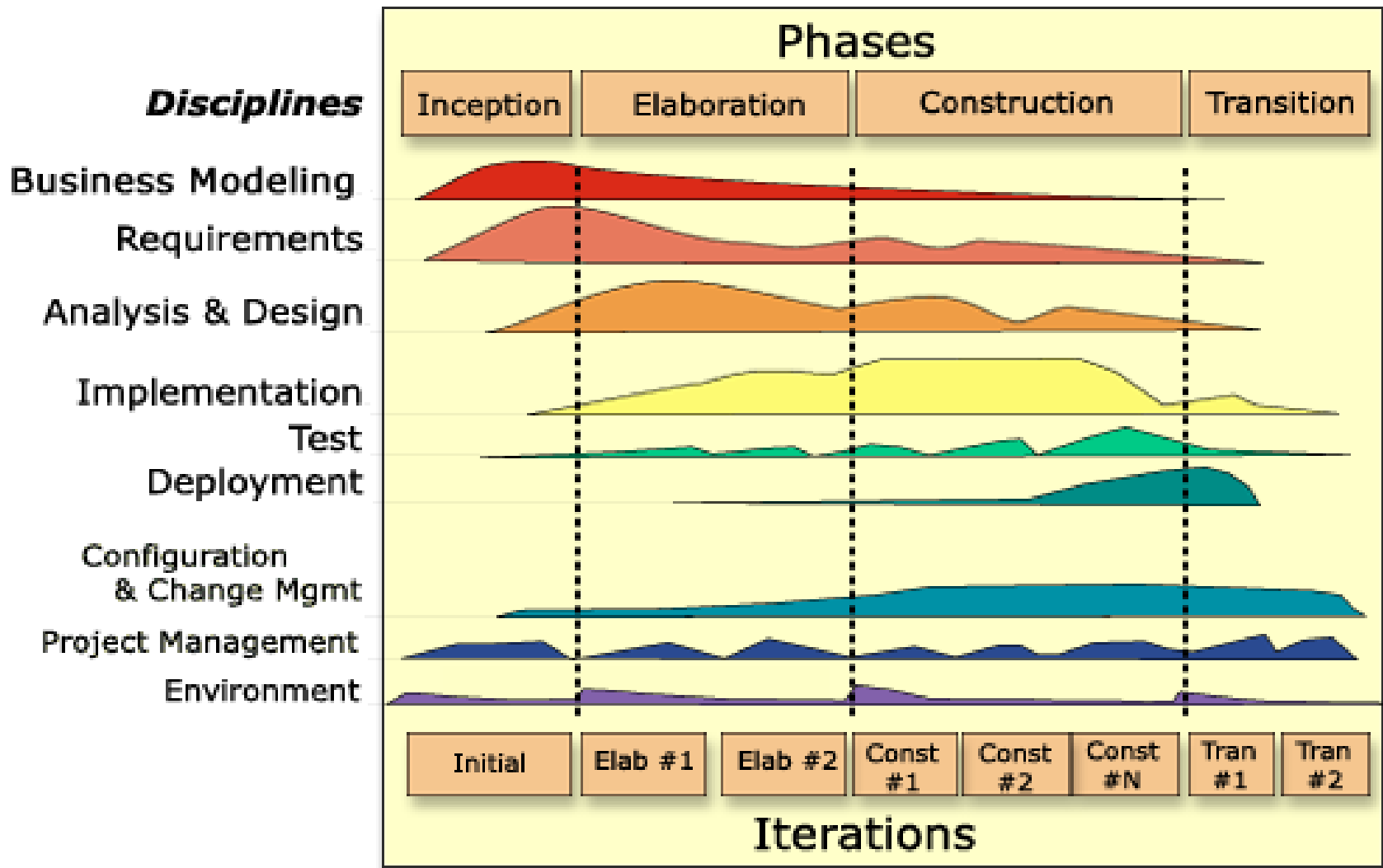
The Rational Unified Process

- A modern generic process derived from the work on the UML and associated process.
- Brings together aspects of the 3 generic process models discussed previously.
- 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.

Phases in the Rational Unified Process



Iterative and Incremental Process in RUP



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 iteration

- In-phase iteration
 - Each phase is iterative with results developed incrementally.
- Cross-phase iteration
 - As shown by the loop in the RUP model, the whole set of phases may be enacted incrementally.

Key points

- Processes should include activities to cope with change. This may involve a prototyping phase that helps avoid poor decisions on requirements and design.
- Processes may be structured for iterative development and delivery so that changes may be made without disrupting the system as a whole.
- The Rational Unified Process is a modern generic process model that is organized into phases (inception, elaboration, construction and transition) but separates activities (requirements, analysis and design, etc.) from these phases.

End...

