ICT 2402 Software Engineering

Software Process Models

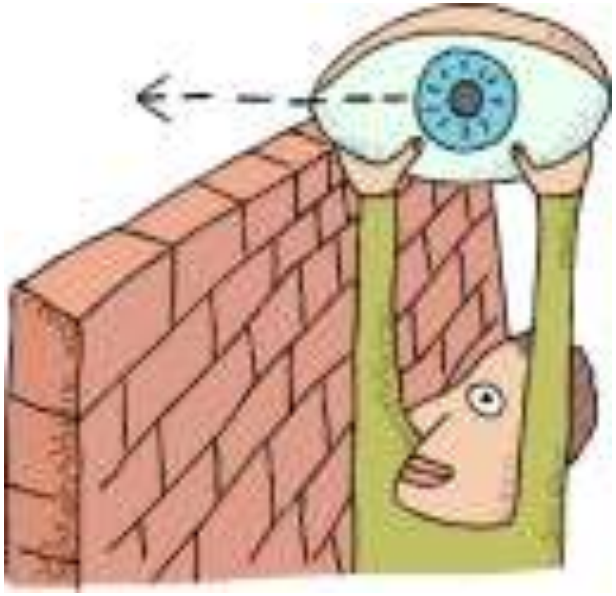
Objectives

After completing this lesson you should be able to

- Categorize different software process models
- Describe three generic process models and when they may be used
- Discuss process iteration
- Describe the process activities in-brief

Topics covered

- Software process models
- Process iteration
- Process activities



The Software Process

- A structured set of activities required to develop a software system
  - Specification
  - Design & Implementation
  - Validation
  - Maintenance

Software Process Model

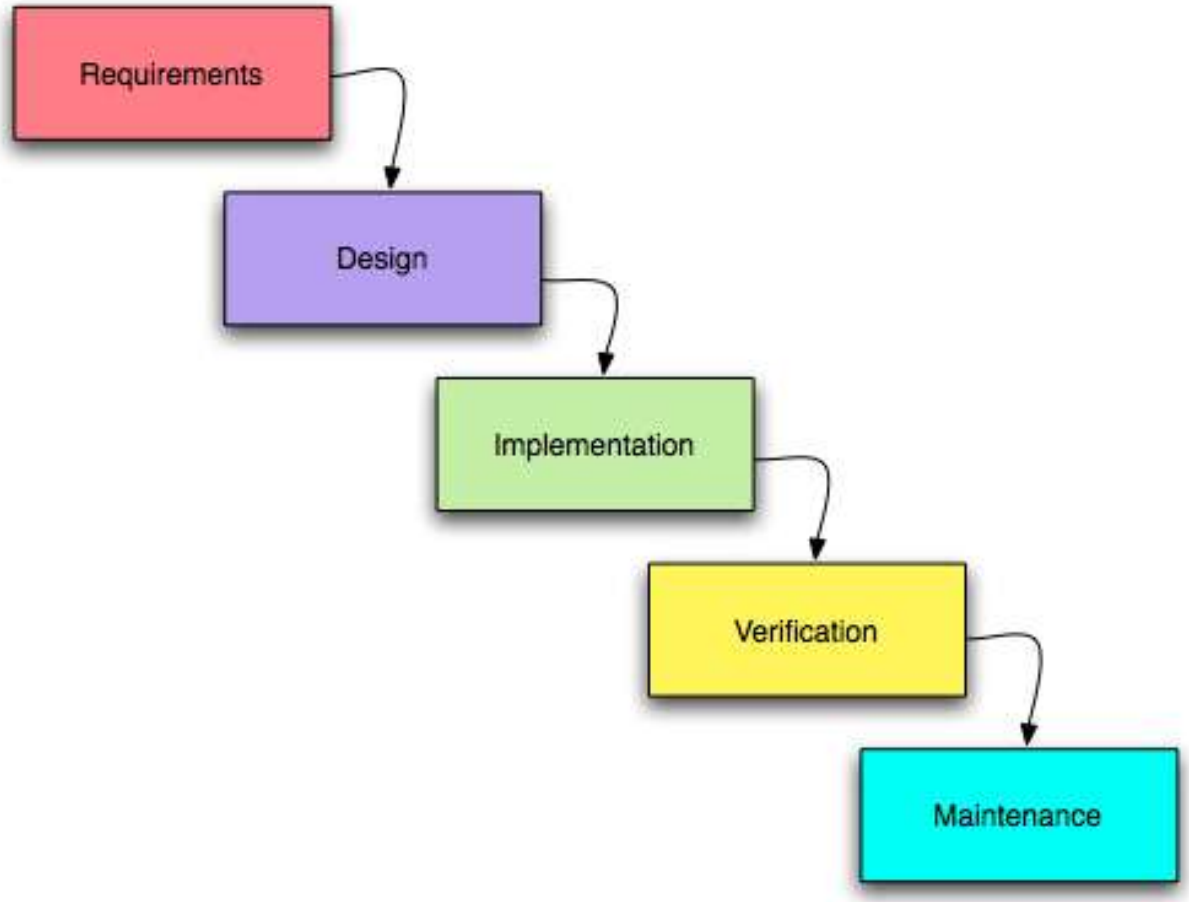
- "A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective"

  - Sommerville

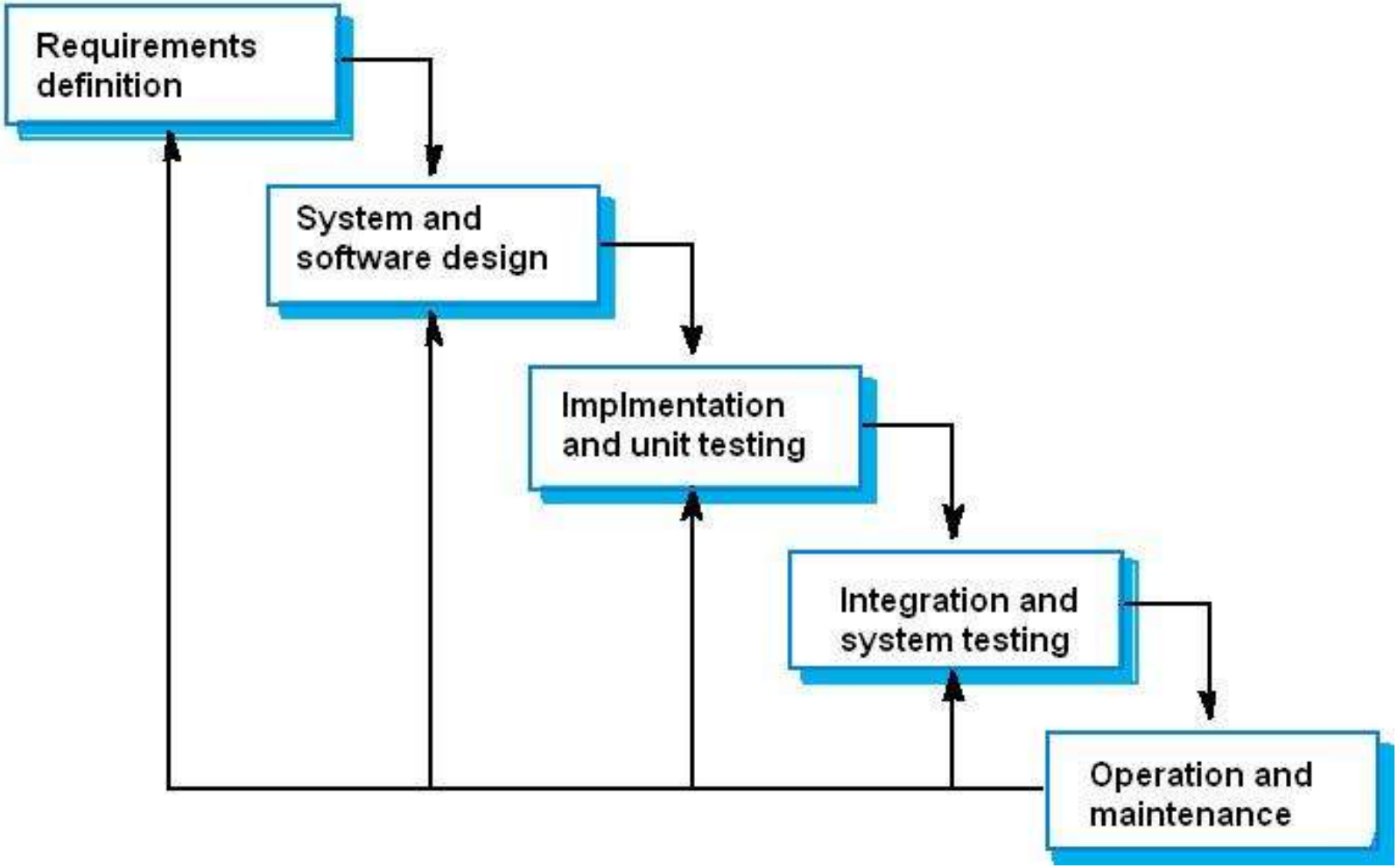
Generic software process models

- The waterfall model
- Iterative processes
- Component-based software engineering

Waterfall Model



Cont...


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

Waterfall model phases

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

Arguments for Waterfall model

- More economical if bugs found earlier
- Better documentation
- Pure waterfall model → highly reliable system
- Widely used (US Dept. of Defense, NASA)

Waterfall model Applicability

- Large systems engineering projects where a system is developed at several sites
- Routine types of projects where requirements are well understood
- When developing organization is familiar with problem domain

Advantages of waterfall model

- Enforce a disciplined approach for software development
- Documents are produced at each stage. This improves the visibility of the project.

Criticism of Waterfall model

- Accommodating change is difficult
- Difficult to respond to changing customer requirements
  - Due to the cost of producing and approving documents, iterations are costly and involves significant rework. Therefore, after a small number of changes, requirements have to be freeze.
- Requirements and limitations cannot be entirely known before completion
- Impossible to get one phase perfected
- Risk management is not covered
- Waste of resources

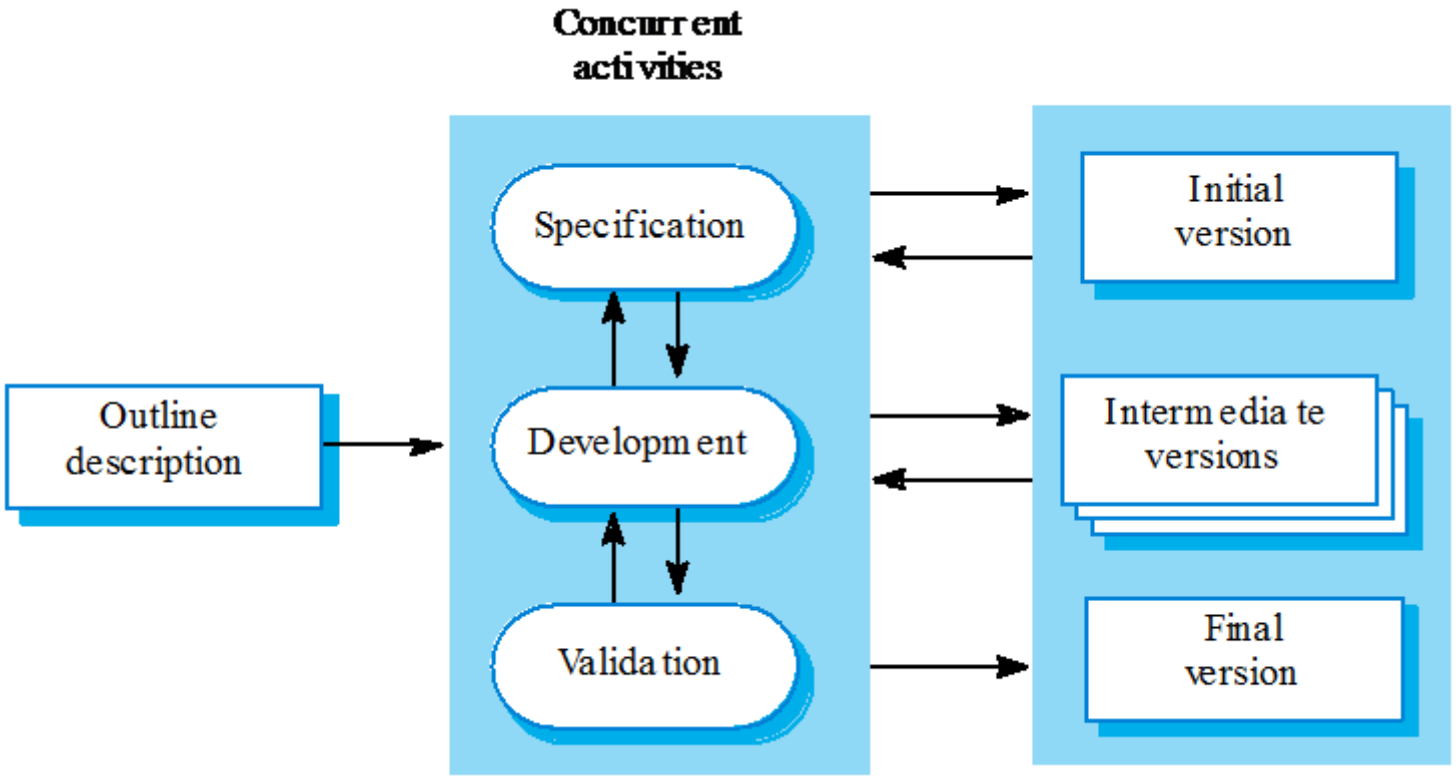
Iterative Processes

- Requirements of a software project tend to change all the time Software process activities may iteratively repeat to cater for the changing requirements
- In iterative development models, requirements specification is developed in conjunction with the software
  - Incremental Development
  - Spiral model
  - Agile software development
  - Extreme programming (XP)
  - Test Driven development (TDD)

Incremental development

- Incremental development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed.
- Specification, development, and validation activities are interleaved.

Incremental development



Incremental development vs waterfall

- The cost of accommodating changing customer requirements is reduced.
- It is easier to get customer feedback on the development work
- More rapid delivery and deployment of useful software that contains the highest priority requirements.

Incremental development advantages

- Customers can use the early increments as prototypes and gain experience that informs their requirements for later system increments. Unlike prototypes, these are part of the real system so there is no re-learning when the complete system is available.
- Customers do not have to wait until the entire system is delivered before they can gain value from it. The first increment satisfies their most critical requirements so they can use the software immediately.
- The process maintains the benefits of incremental development in that it should be relatively easy to incorporate changes into the system.
- As the highest-priority services are delivered first and increments then integrated, the most important system services receive the most testing. This means that customers are less likely to encounter software failures in the most important parts of the system.

Incremental development disadvantages (management view)

- 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.

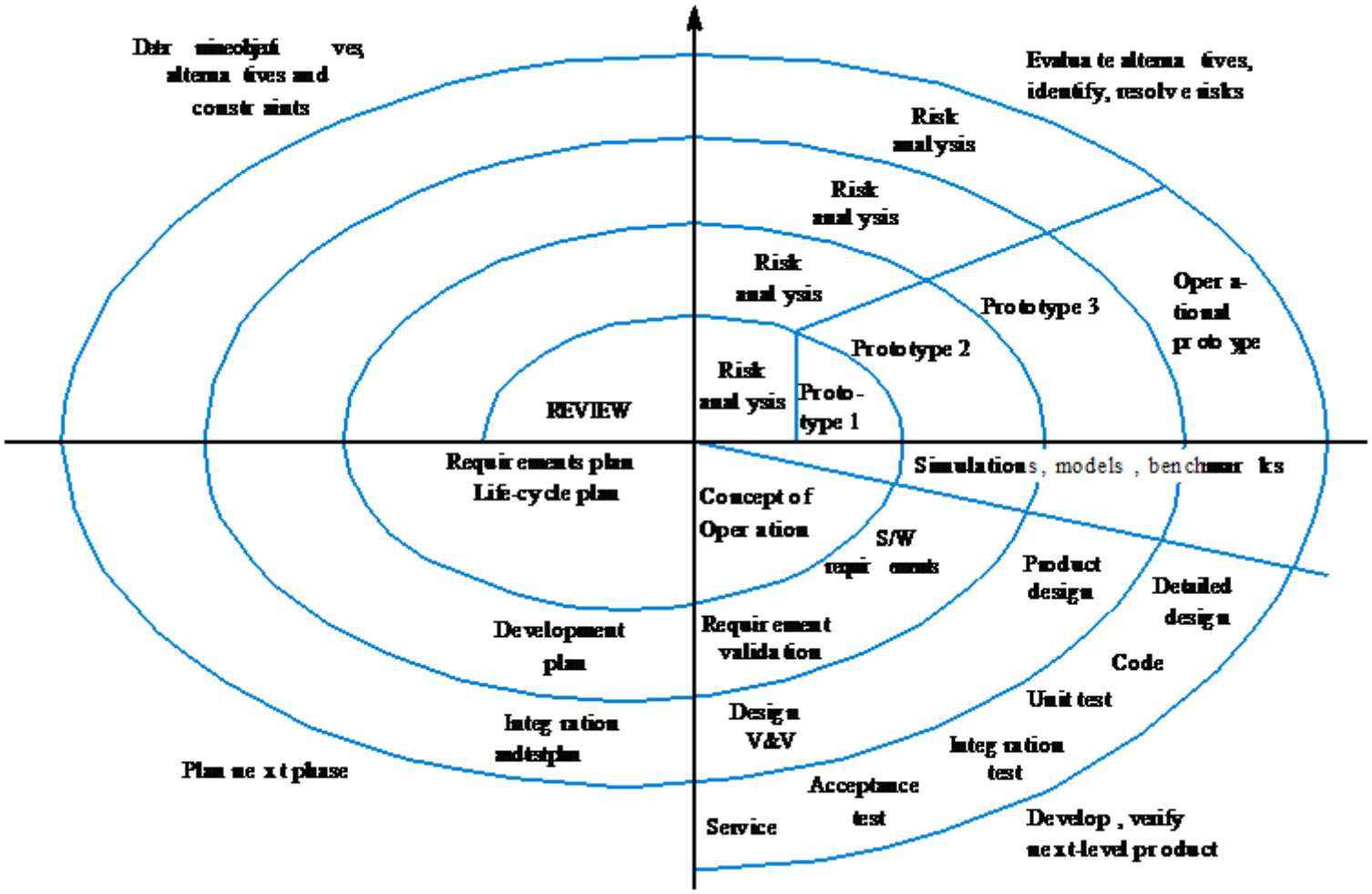
Incremental development disadvantages

- 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.
- Iterative development can also be difficult when a replacement system is being developed. Users want all of the functionality of the old system and are often unwilling to experiment with an incomplete new system. Therefore, getting useful customer feedback is difficult.
- 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. In the incremental approach, there is no complete system specification until the final increment is specified. This requires a new form of contract, which large

Spiral development

- Favored for large, expensive, and complicated projects
- Process represented as a spiral rather than as a sequence of activities with backtracking
- Each loop in the spiral is a phase in the process
- No fixed phases - loops in the spiral are chosen depending on what is required
- Risks are explicitly assessed and resolved throughout the process

Spiral model



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

Agile software development

- Phases:
  - Planning
  - requirements analysis
  - Design
  - Coding
  - Testing
  - documentation
- Developing software in short timeboxes
  - called iterations (one to four weeks)
- Software is released in mini-increments at the end of each iteration

Agile software development

- At the end of each iteration, the team re-evaluates project priorities
- Emphasize working software as the primary measure of progress
- Emphasize real-time communication (bullpen), preferably face-to-face, over written documents

Agile vs. Iterative

- Time period is measured in weeks rather than months
- Work is performed in a highly collaborative manner
- Treating time period as a strict timebox

Agile vs. Waterfall

- Produce completely developed and tested features within a phase
- Emphasis is on obtaining the smallest workable piece of functionality
- Continually improving/adding further functionality

Prototyping

- Begins with requirements collection, followed by prototyping and user evaluation
- Prototypes are built based on feedback from users
- Help users get an idea of the system
- Make it easier for users to make design decisions without waiting for the system to be built

Drawbacks in Prototyping

- Once the users see the prototype, difficult to convince that the finished design will not be produced for some time
- Using patched-together prototype code in the real system
- Designers and end users can focus too much on user interface design

Cowboy coding

- Without an actual defined method – team members do whatever they feel is right
- Often involve one programmer

Advantages of Cowboy coding

- Allows for quick solutions for small problems
- A problem is small enough and well understood
- Involve only a single developer
- Can allow a 'spike' to see if a programming idea is valid before embarking on a larger project that depends on the idea

Disadvantages of Cowboy coding

- Lacks a clear scope/vision
- Suitable for small projects
- Tends to produce poorer quality/buggy software
- Lack of version control
- Huge risk to the business

Extreme programming

- Stresses customer satisfaction
- Empowers developers to confidently respond to changing customer requirements, even late in the life cycle
- Get feedback by testing their software starting on day one

Extreme programming

- Enable groupware style development
- Improves a software project in five essential ways; communication, simplicity, feedback, courage and respect
- Deliver the system as early as possible and implement changes as suggested

XP activities

- Coding
  - drawing diagrams that generate code, scripting a web-based system, coding a program to be compiled
- Testing
  - uses Unit Tests - automated tests to test the code
  - uses acceptance tests - based on requirements given by the customer

XP activities

- Listening
  - Programmers have to listen to what the customer needs
  - give the customer feedback
  - improve the customer's own understanding the problem
- Designing
  - creating a design structure that organizes the logic in the system
  - changing one part of the system will not affect other parts of the system

XP applicability

- For research projects focusing on domain knowledge
- For projects involve new or prototype technology, where the requirements change rapidly

Test Driven Development (TDD)

- A software development technique
- Repeatedly first writing a test case and then implementing only the code necessary to pass the test
- Gives rapid feedback
- A method of designing software, not merely a method of testing

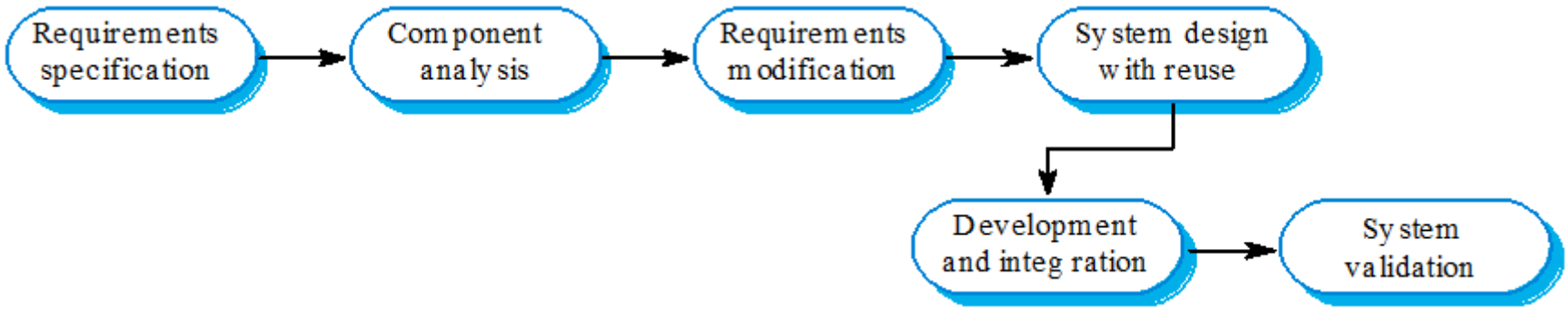
Test-Driven Development Cycle

- Add a test
- Run all tests and see the new one fail
- Write some code
- Run the automated tests and see them succeed
- Refactor code

Component-based software engineering

- Systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems
- Need a large base of reusable software components and an integrating framework`
- Process stages
  - Component analysis
  - Requirements modification
  - System design with reuse
  - Development and integration

Reuse-oriented development



Advantages of Component Based Software Engineering

- Reduce the amount of software to be developed
- Reduce cost
- Reduce risk
- Faster delivery of software

Disadvantages of Component Based Software Engineering

- Requirements have to be compromised
  - There may not be suitable software components to match exact client requirements. Therefore certain requirements may be compromised.
- Evolution of the software not fully in control
  - When third-party software components are used, modifications or upgrades to the component are done by the third party. So the developing organization does not have full control over the evolution of the software

Questions?

References

- Chapter 2 & 3 Software Engineering – 9th Edition

  Ian Sommerville