Software Engineering

Roger S.Pressman , Ph.D.

Computer science and engineering

Team 1.
200711440
Song Tae soo

1. Software Engineering

- **Definition of software engineering:**

  Software Engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.
- Software Engineering is a **layered technology.**

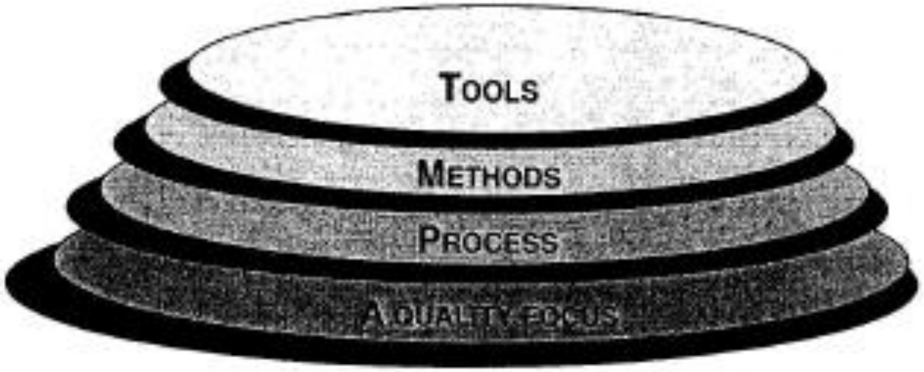


Figure 1 Software engineering layers.

2. A Process Framework

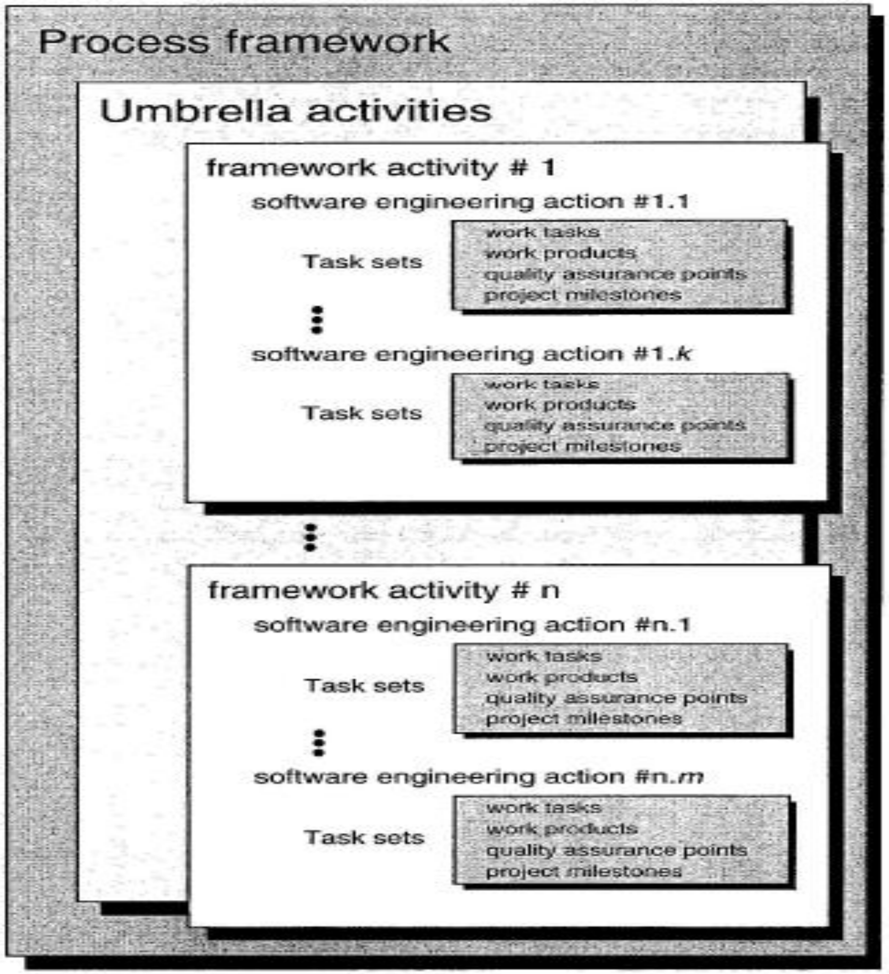


Figure 2 A software process framework.

- **A generic process framework**
  - –Communication
  - –Planning
  - –Modeling
  - –Construction
  - –Deployment

- The details of the software process will be quite different in each case, but **the framework activities** remain the **same.**

- **Typical activities**
  –Software project tracking and control
  –Risk management
  –Software quality assurance
  –Formal technical reviews
  –Measurement
  –Software configuration management
  –Reusability management
  –Work product preparation and production

3. Software Process Models

- “Where we locate the basis when we choose one of that, Process Models?”

- 3.1. Prescriptive Models

    3.1.1. The waterfall model

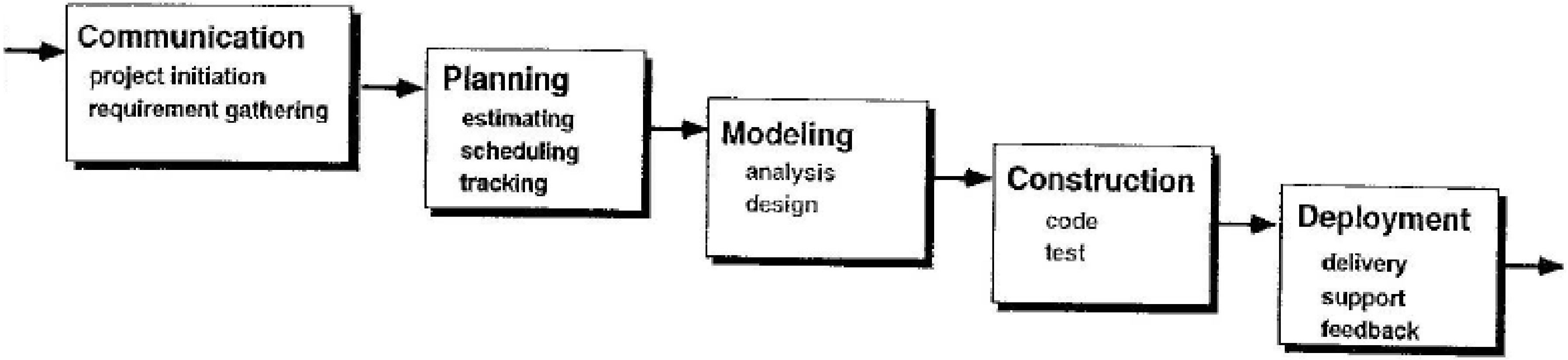
    3.1.2. Incremental process models

    3.1.3. Evolutionary process models

    3.1.4. Specialized process models

    3.1.5. The unified process

3.1.1. The waterfall model



Communication
project initiation
requirement gathering

Planning
estimating
scheduling
tracking

Modeling
analysis
design

Construction
code
test

Deployment
delivery
support
feedback

Figure 3 The waterfall model

3.1.2. Incremental process models

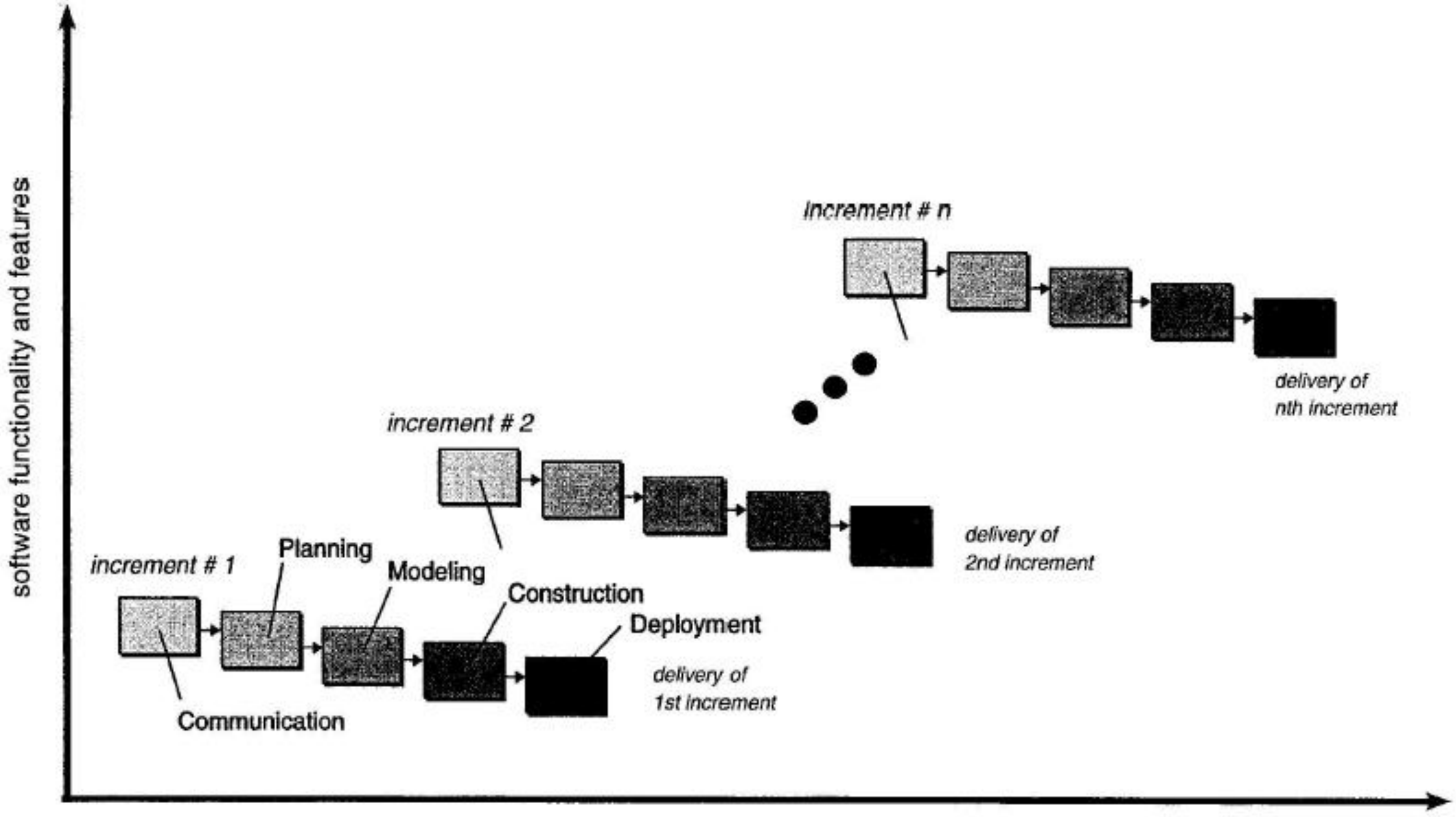


Figure 4 The incremental model.

3.1.3. Evolutionary process models

- Prototyping Paradigm
- The Spiral Model

Prototyping

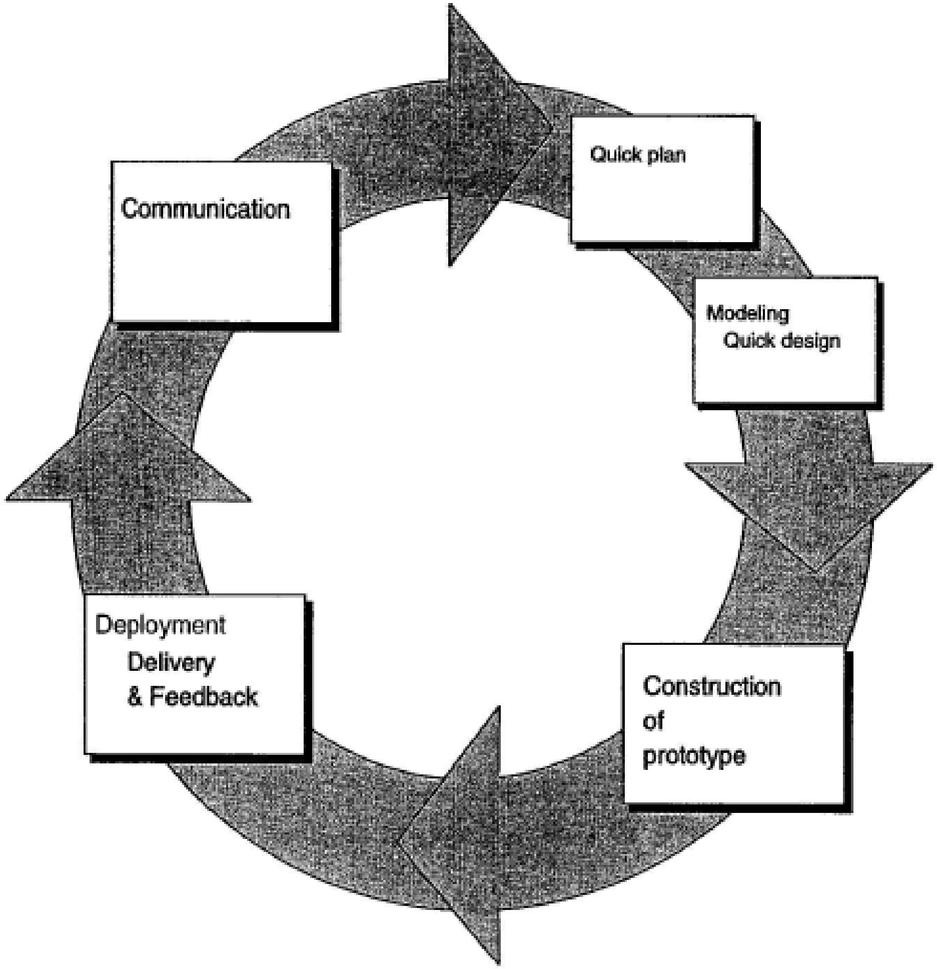


Figure 5 The prototyping paradigm.

The Spiral Model

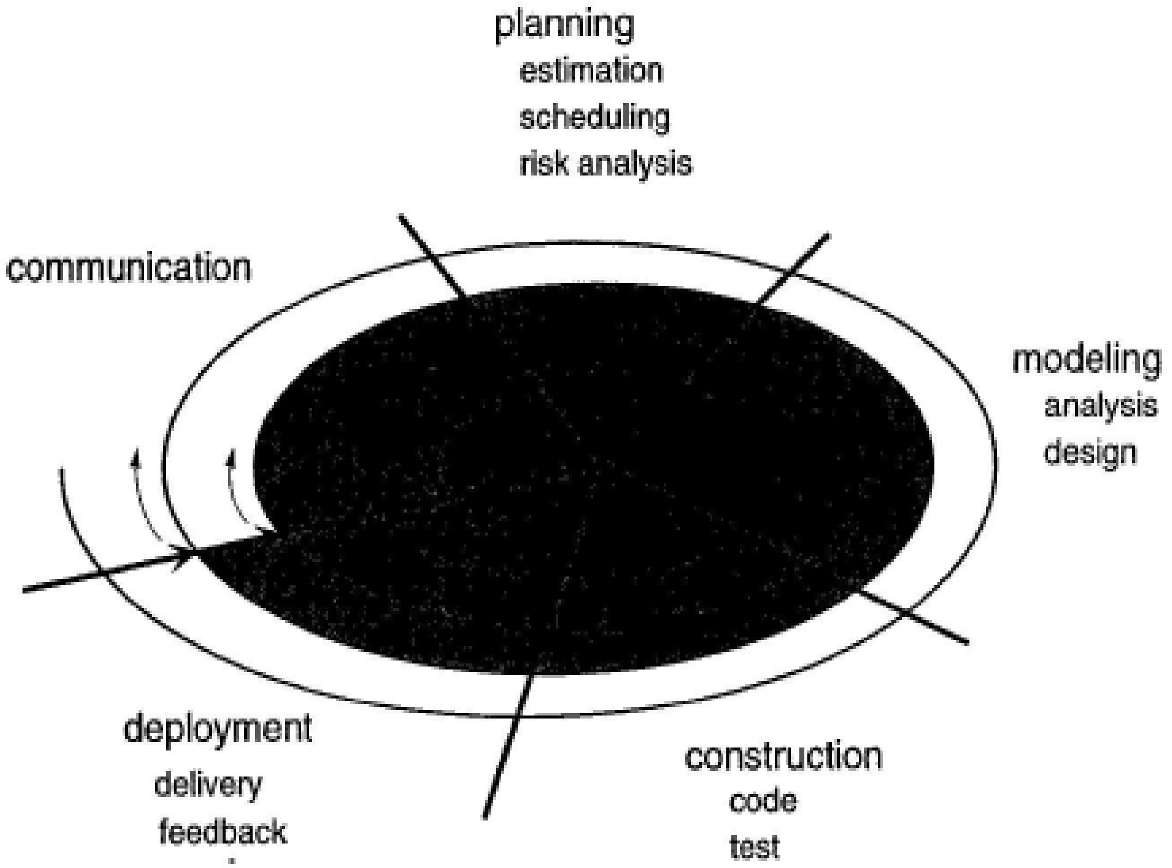


Figure 6 A typical spiral model.

3.1.4. Specialized process models

- Component-Based Development
- The Formal Methods Model
- Aspect-Oriented Software Development

3.1.5. The Unified Process

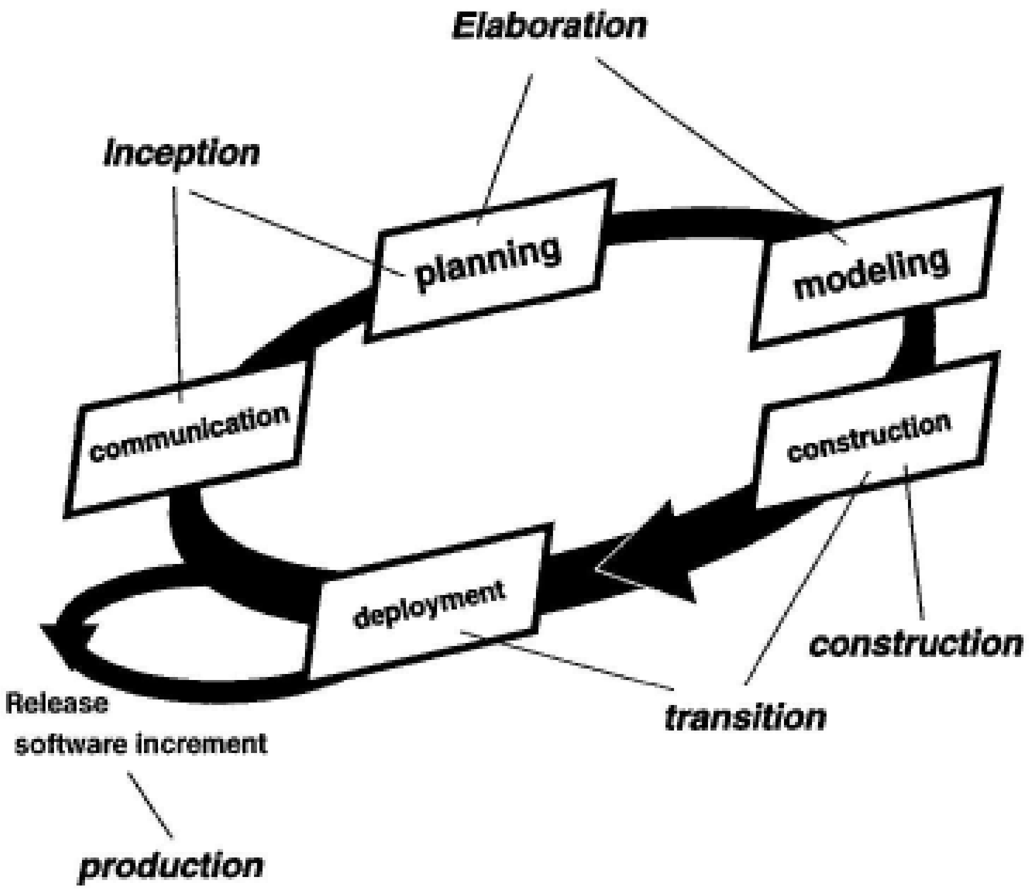


Figure 7 The Unified Process (UP).

3.2. Agile Software Development

- Individuals and interactions
  over processes and tools
- Working software
  over comprehensive documentation
- Customer collaboration
  over contract negotiation
- Responding to change
  over following a plan

3.2. Agile Software Development

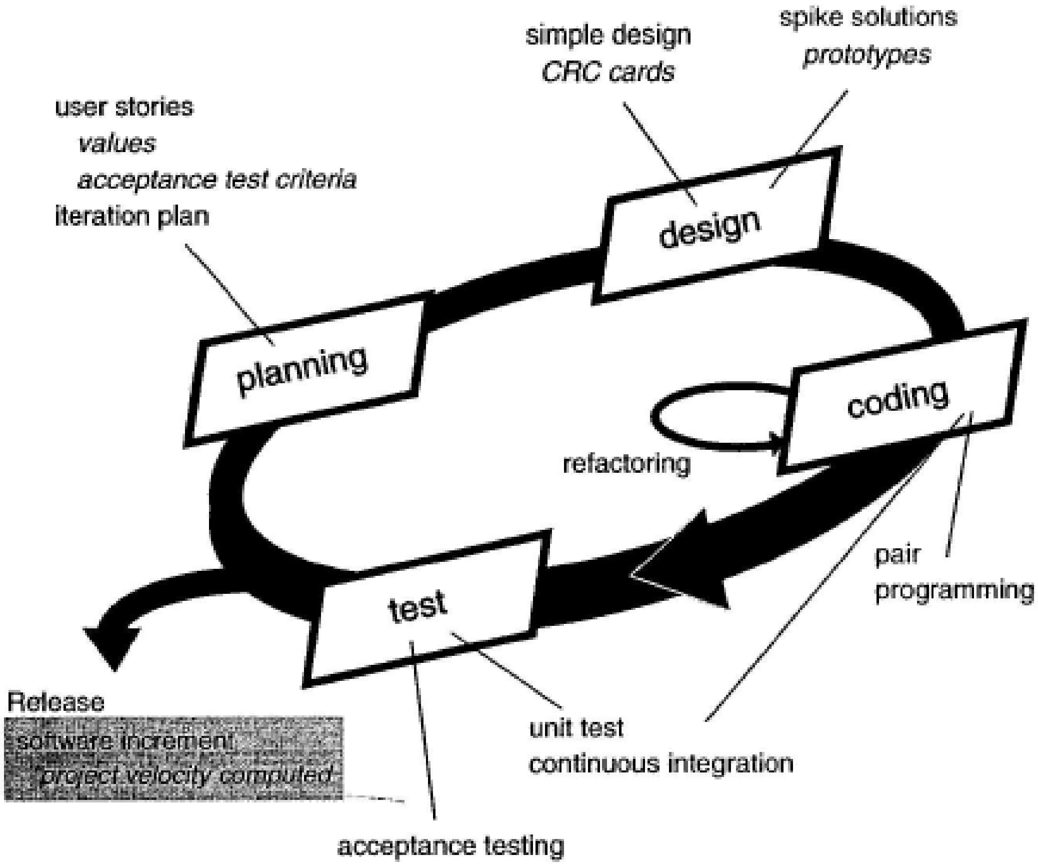


Figure 8 The Extreme Programming process.

4. The Management Spectrum

- Effective software project management focuses on the 3P: **PEOPLE, PROBLEM, PROCESS.**

  -PEOPLE

  The people management maturity model

  -PROBLEM

  The software developer and customer must meet to define project objectives and scope.

  →Joint Application Design(JAD)

  -PROCESS

  The Capability Maturity Model Integration(CMMI)

  (1)"continuous" model
  (2)"staged" model

5.Software Project Management

- **Measurement and Metrics**

- **Project Estimating**

  Three broad classes of estimation techniques for software projects:

  1. Effort estimation techniques
  2. Size-Oriented Estimation
  3. Empirical Models

- **Risk Analysis**

  The goals of risk analysis:

  (1) to identify those risks that have high likelihood of occurrence.

  (2) to assess the consequence of each risk should it occur.

  (3) to develop a plan for mitigating the risks when possible, monitoring factors that may indicate there arrival, and developing a set of contingency plans should they occur.

- **Scheduling**

- **Tracking and Control**

  Control focuses on two major issues: Quality and Change.

6. Software Quality Assurance

- Software requirements are the foundation from which quality is assessed.
- A mature software-process model defines a set of development
  criteria that guide the manner in which software is engineered.
- There is a set of implict requirements that often goes unmentioned(e.g. the desire for good maintainability).

A Set Of Quality Factors

- Correctness.
- Reliability.
- Efficiency.
- Integrity.
- Usability.
- Maintainability.
- Flexibility.
- Testability.
- Portability.
- Reusability.
- Interoperability.

7. Software Configuration Management

- Software configuration management (SCM) is an umbrella activity that is applied throughout the software process.

- SCM activities are developed to

  (1) identify changes

  (2) control changes

  (3) ensure that changes are being properly implemented

  (4) report changes to others who may have an interest.

- A primary goal of software engineering

  → to improve the ease with which changes can be accommodated and reduce the amount of effort expended when changes must be made.

8. The Technical Spectrum

- 8.1 Software Engineering Methods
  - The Landscape.
- 8.2 Problem Definition.
- 8.3 Design.
- 8.4 Program Construction.
- 8.5 Software Testing.

8.1 Software Engineering Methods – The Landscape.

- All engineering disciplines encompass four major activities.

(1) the definition of the problem to be solved

(2) the design of a solution that will meet the customer's needs.

(3) the construction of solution.

(4) the testing of the implemented solution.

- The methods landscape's three different methods.

(1)Conventional software engineering methods.

(2)Object-oriented approaches.

(3)Formal methods.

8.2 Problem Definition

- 8.2.1 Analysis Principles
- 8.2.2 Analysis Methods

8.2.1 Analysis Principles

- 1. The data domain of problem must be modeled.
- 2. The functional domain of the problem must be modeled.
- 3.The behavior of the system must be represented.
- 4.Models of data, function, and behavior must be partitioned.
- 5.The overriding trend in analysis is from essence toward implementation.

8.2.2 Analysis Methods

- All analysis methods provide a notation for describing data objects and the relationships that between them.
- All analysis methods couple function and data and provide a way to understand how function operates on data.
- All analysis methods enable an analyst to represent behavior at a system level and, in some cases, at a more localized level
- All analysis methods support a partitioning approach that leads to increasingly more detailed and implementation-specific models.
- All analysis methods establish a foundation from which design begins, and some provide representations that can be directly mapped into design.

Analysis Model

- 1. Scenario-based elements.
- 2. Class-based elements.
- 3. Behavioral elements.
- 4. Flow-oriented elements.

8.3 Design

- 8.3.1 Design Principles

- 8.3.2 The design Pyramid

8.3.1 Design Principles

- 1. Data and the algorithms that manipulate data should be created as a set of interrelated abstractions
- 2. The internal design detail of data structures and algorithms should be hidden from other software components that make use of the data structures and algorithms.
- 3. Modules should exhibit independence.
- 4. Algorithms should be designed using a constrained set of logical constructs.

8.3.2 The design Pyramid

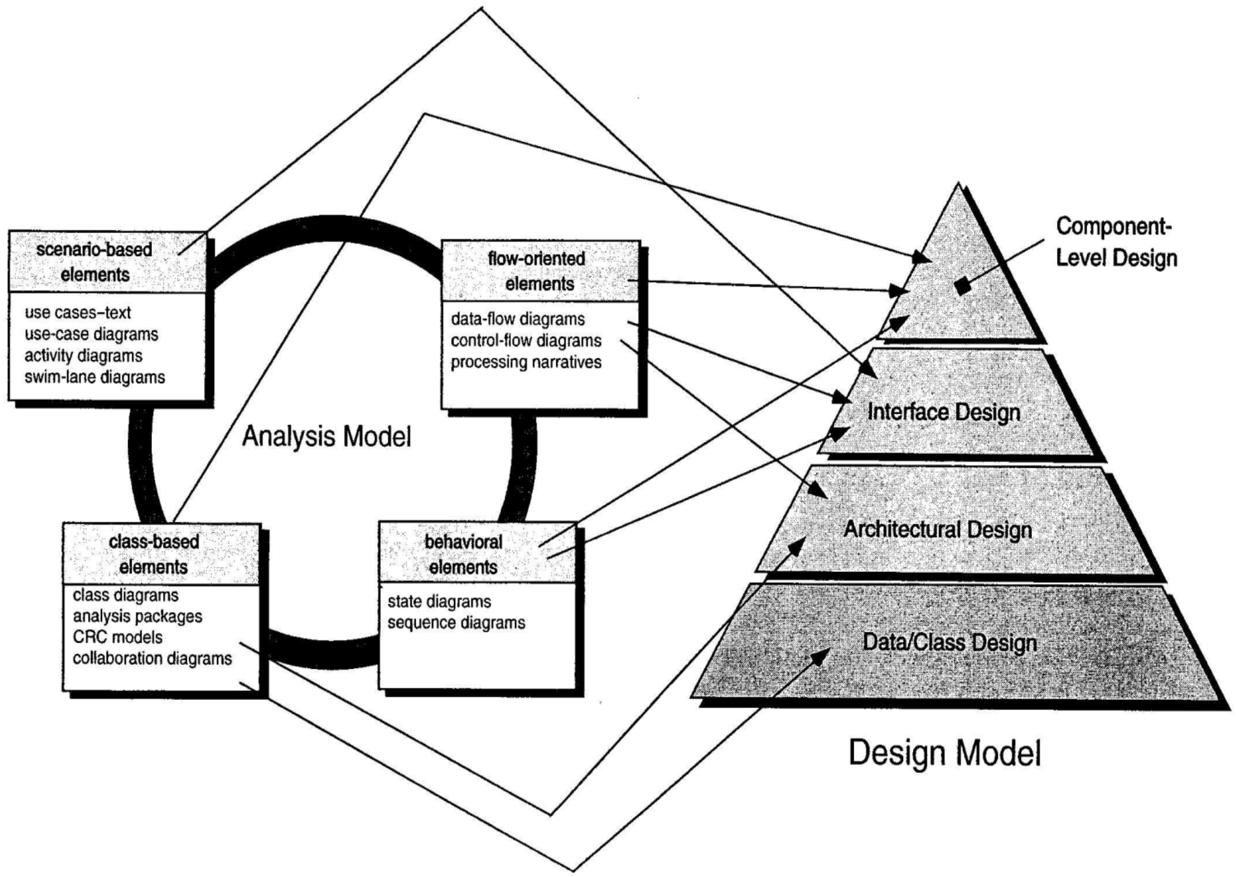


Figure 9 The design pyramid.

8.4 Program Construction

- Best programming language?
- Computer-based system < construction
- Innovative approaches to analysis and design
- Comprehensive SQA techniques
- Effective and Efficient testing

8.5 Software Testing

- Glen Myers's rules.

- 1.Testing is a process of executing a program with the intent of finding an error.
- 2. A good test case is one that has a high probability of finding an as-yet-undiscovered error.
- 3. A successful test is one that uncovers an as-yet-undiscovered error.

8.5 Software Testing

- 8.5.1 Strategy

- Testing begins at the module level and works incrementally “outward” toward the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points in time.
- Testing is conducted by the developer of the software and (for large projects) an independent test group.
- Testing and debugging are different activities, but debugging must
  be accommodated in any testing strategy.

- 8.5.2 Tactics
  Black-box testing
  White-box testing

9.Software Engineering Patterns

- Software process -> collection of patterns.
- provides us with a template.
- any level of abstraction.
- Process , analysis , design , testing patterns .

9.1 process patterns

- Effective mechanism for describing any software process.
- High level of abstraction -> hierarchical process description.

9.2 analysis patterns

- Reoccur across all project within a specific application.
- Integrated into the analysis model by reference to the pattern name.

9.3 design patterns

- Provide a description that enables a designer to determine

  1) whether the pattern is applicable to the current work.
  2) whether the pattern can be reused.
  3) whether the pattern can serve as a guide for developing a similar but functionally or structurally different pattern.

- Design patterns are…

  Pattern name, intent, also known as ,motivation ,applicability, structure, participants ,collaborations , consequences, related patterns

10. The Road Ahead And The Three Rs

- Staff downsizing
- Growing reality of international outsourcing
- Revolution of software engineering
- Three Rs

  1)Reuse – risk, cost, revenue => best hope!

  2)Reengineering – long time => step by step!

  3)Retooling – reuse & reengineering => Up!

11. Summary

- Will we continue to struggle to produce software that meets the needs of a new breed of customers?
- Will software remain a bottleneck in the development of new generations of computer-based products and systems?
- The degree to which industry embraces software engineering
- culture of software development
- We should look to the future with anticipation or trepidation.