Software Engineering - Course Overview

Dr.-Ing. Ali Diab

Outline

- What Is Software?
- Why Is Software Engineering So Important?
- Goals
- Organizational Stuff
- How to Handle the Course

What Is Software?

- Software:
  - The product that software professionals build and then support over the long term

- Software encompasses
  - Instructions (computer programs)
  - Data structures that enable the programs to adequately store and manipulate information
  - Documentation that describes the operation and use of the programs

Why Is Software Engineering So Important?

- The economies of ALL developed nations are dependent on software
- More and more systems are software controlled (transportation, medical, telecommunications, military, industrial, entertainment, etc.)
- Maintaining software is cost intensive → **software engineering**
  - Software engineering is concerned with theories, methods and tools for professional software development

Goals

- Provision of up-to-date knowledge concerning Software engineering techniques

- In-depth understanding of how to solve technical problems in this concern

- Focus on
  - Software development
    - Requirement, performance, methods, etc.
  - Software engineering cycle
  - Software management
  - Service-oriented software engineering
  - ...

Organizational Stuff

- Lecturer contact information
  - Dr.-Ing. Ali Diab
  - email: adiab@albaath-univ.edu.sy (Subject: Software Engineering)

- Course prerequisites
  - Basics in programming
  - Good basics in Mathematic

- Course budget
  - 1 Lecture per week

How to Handle the Course

- Cover your knowledge holes
  - Go through basics

- Go through the material provided
  - Will be electronically provided
  - References will be listed for each lecture
    - If possible, electronic copy will be provided
  - Search the Internet for knowledge if required

- Questions catalog will be provided
  - Cover 50 – 60 % of the course

- Exam
  - ?????

Software Engineering

-

Software & Software Engineering

Dr.-Ing. Ali Diab

Outline

- Introduction
- Definition & Importance
- Layered Technology
- Software Engineering Process
- Software Engineering Costs

Introduction

Software Products

- Generic products
  - Stand-alone systems
    - Marketed and sold to any customer who wishes to buy them
  - Examples
    - PC software such as editing, graphics programs, project management tools; CAD software, etc.

- Customized products
  - Software products commissioned by a specific customer to meet their own needs
  - Examples
    - Embedded control systems, air traffic control software, traffic monitoring systems, etc.

Why Is Software Engineering So Important?

- The economies of ALL developed nations are dependent on software
- More and more systems are software controlled (transportation, medical, telecommunications, military, industrial, entertainment, etc.)
- Maintaining software is cost intensive $\rightarrow$ **software engineering**
  - Software engineering is concerned with theories, methods and tools for professional software development

Software Costs

- Software costs often dominate computer system costs
  - The costs of software on a PC are often greater than the hardware cost

- Software costs are in their majority maintaining costs rather than developing
  - For systems with a long life, maintenance costs may be several times development costs

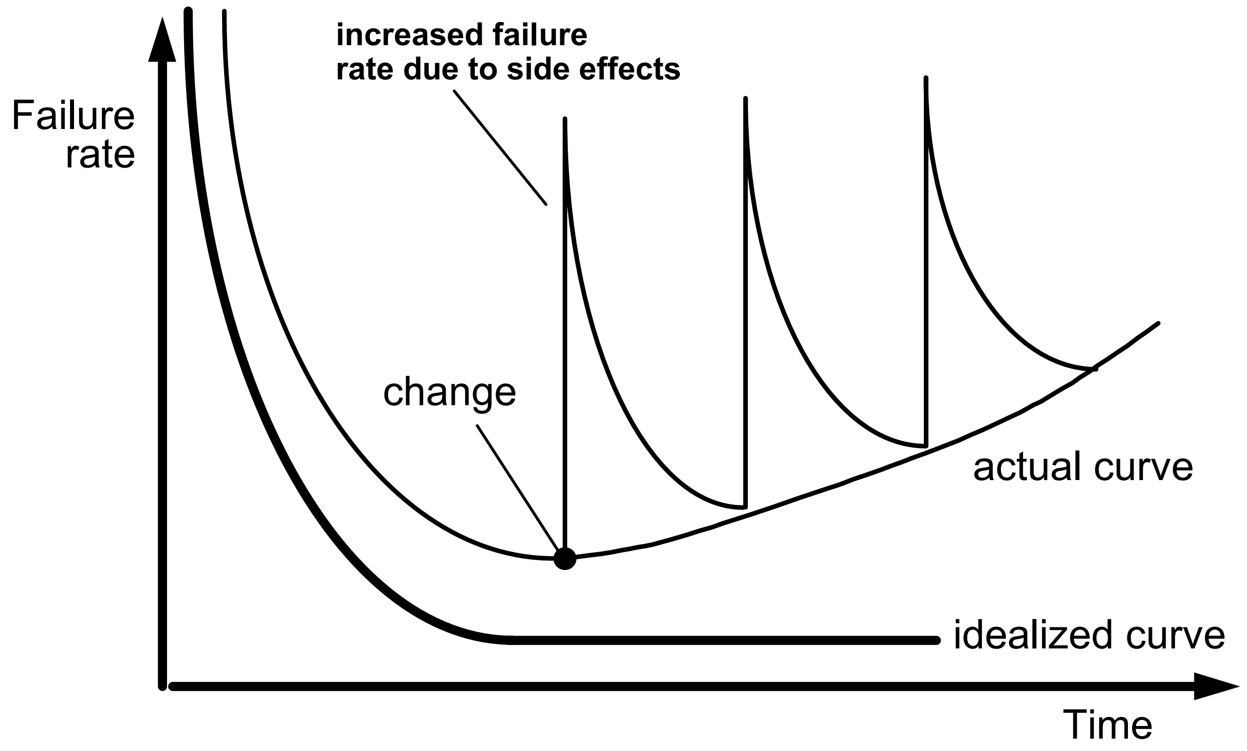
- Software engineering is concerned with cost-effective software development
  - Reduce development costs
  - Reduce maintenance costs
  - **Circumstantial**: the performance of software products themselves is also improved

Features of Software?

- The characteristics of software products make them different from other things human being build
- They are logical system with following features
  - Software is developed or engineered
    - It is not manufactured in the classical sense which has quality problem
  - Software doesn't "wear out” but it deteriorates (due to a certain change)
  - Although the industry is moving toward component-based construction (e.g. standard screws and off-the-shelf integrated circuits), most software continues to be custom-built
    - Modern reusable components encapsulate data and processing into software parts to be reused by different programs. E.g. graphical user interface, window, pull-down menus in library, etc.

Features of Software?

- Hardware has bathtub curve of failure rate (high failure rate in the beginning, then drop to steady state, then cumulative effects of dust, vibration, abuse occurs).



Wear vs. Deterioration

Software Applications

- System software
  - Compilers, editors, file management utilities, etc.
- Application software
  - Stand-alone programs for specific needs
- Engineering/scientific software
  - Characterized by “number crunching” algorithms, such as automotive stress analysis, molecular biology, orbital dynamics, etc.
- Embedded software
  - Resides within a product or system
    - Key pad control of a microwave oven, digital function of dashboard display in a car, etc.

Software Applications

- Product-line software
  - Focus on a limited marketplace to address mass consumer market
    - Word processing, graphics, database management, etc.

- Web applications
  - Network centric software
    - As web 2.0 emerges, more sophisticated computing environments are supported, integrated with remote database and business applications

- <u>AI software uses non-numerical algorithm to solve complex problem</u>
  - Robotics, expert system, pattern recognition game playing, etc.

Software - New Categories

- Open world computing
  - Pervasive, ubiquitous, distributed computing due to wireless networking
    - How to allow mobile devices, personal computer, enterprise system to communicate across vast network
- Netsourcing
  - The Web as a computing engine
    - How to architect simple and sophisticated applications to target end-users worldwide
- Open source
  - ”Free ” source code open to the computing community

Definition & Importance

Definition & Importance

- Definition
  - Seminal definition
    - The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines
  - IEEE definition
    - The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

- Importance
  - More and more, individuals and societies rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly
  - It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project (The majority of costs are the costs of changing the software after it has gone into use)

Terminology

Question	Answer
What is software?	Computer programs, data structures and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

Terminology

Product characteristic	**Description**
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

What are Software Engineering Methods?

- Structured approaches to software development which include system models, notations, rules, design advice and process guidance
  - Model descriptions
    - Descriptions of graphical models which should be produced
  - Rules
    - Constraints applied to system models
  - Recommendations
    - Advice on good design practice
  - Process guidance
    - What activities to follow

What is Computer-Aided Software Engineering

- Computer-Aided Software Engineering (CASE): Software systems intended to provide automated support for software process activities

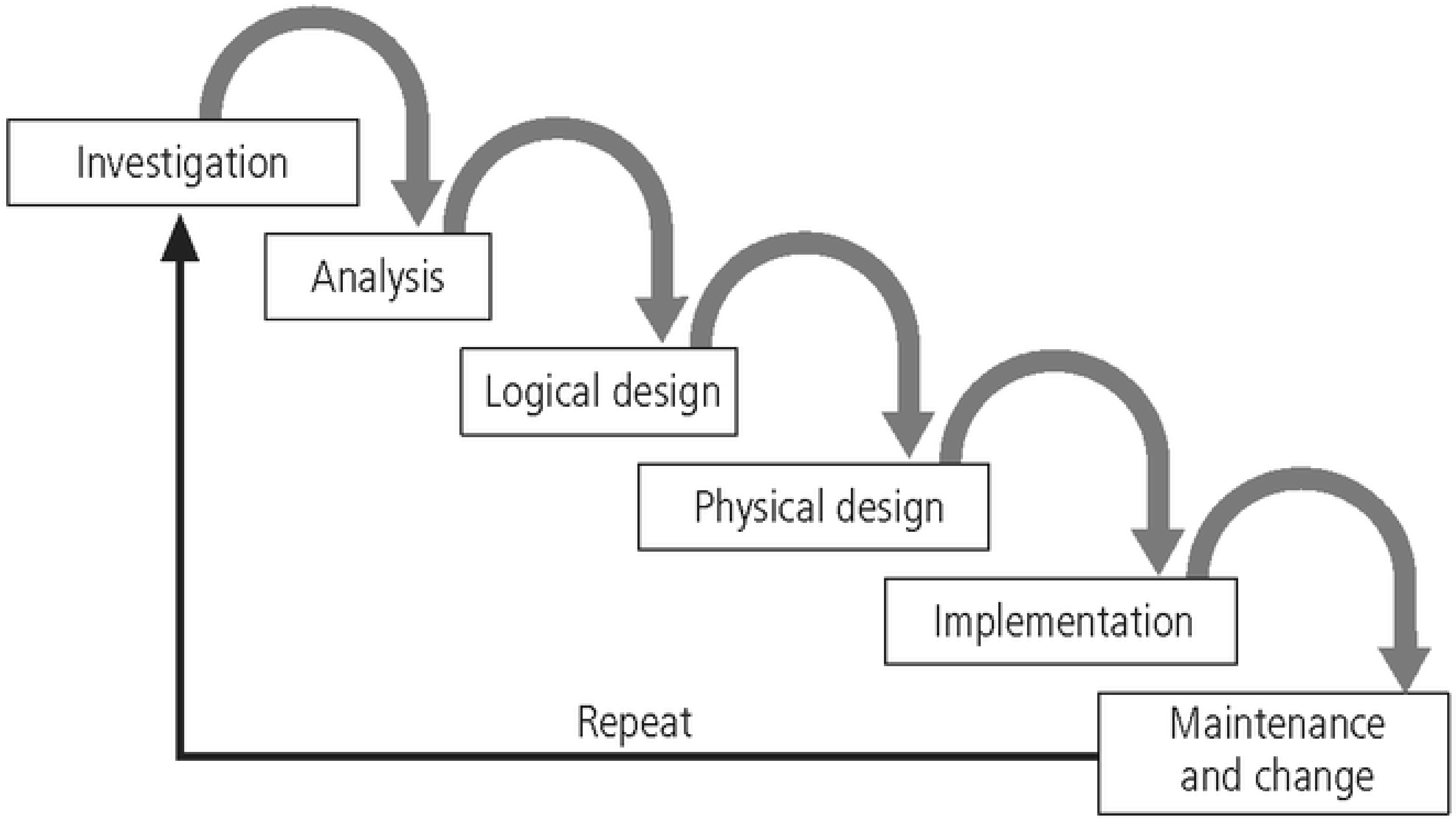
- CASE systems are often used for method support

- Upper-CASE
  - Tools to support the early process activities of requirements and design

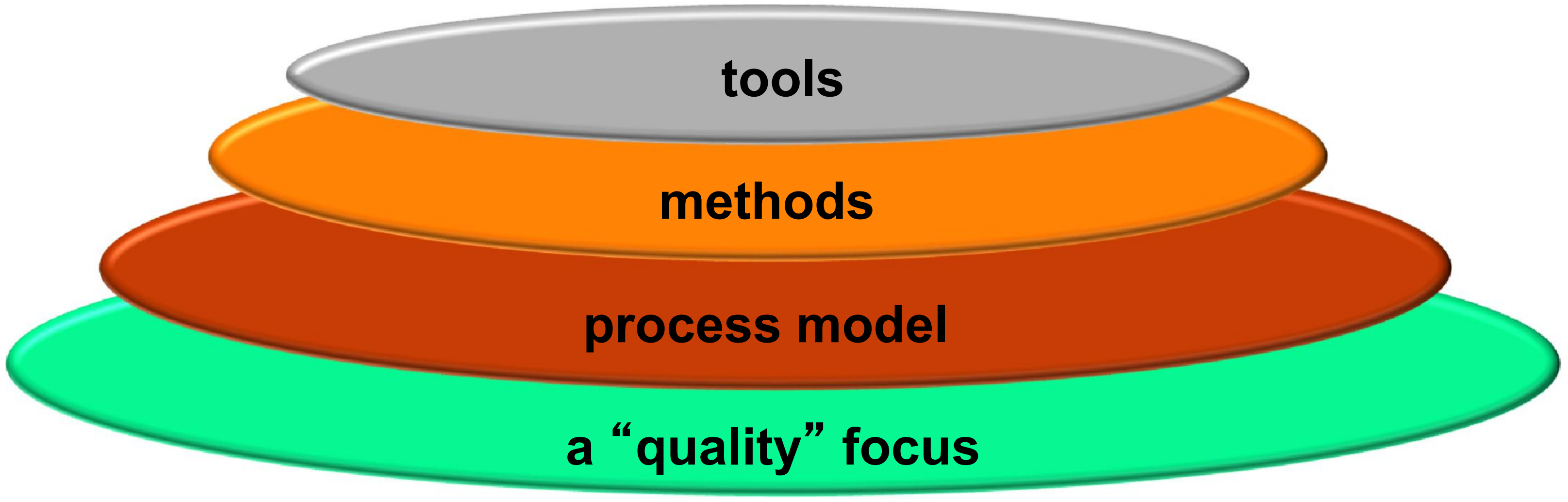
- Lower-CASE
  - Tools to support later activities such as programming, debugging and testing

Layered Technology

Systems Development Life Cycle (SDLC)

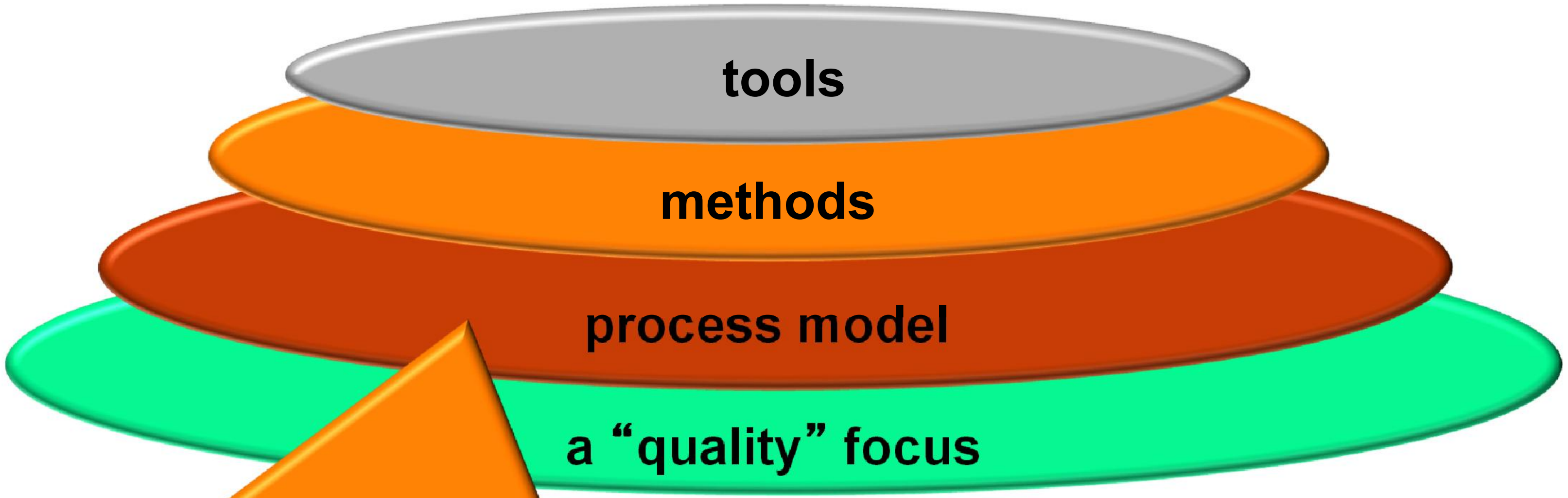


Software Engineering - A Layered Technology



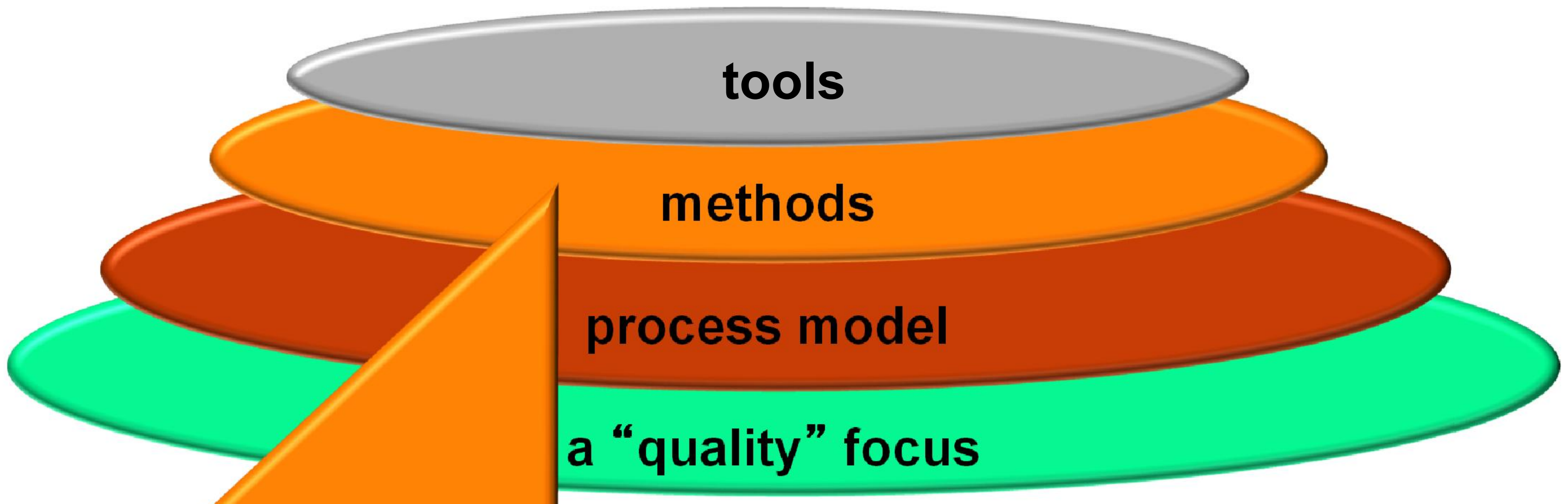
- Any engineering approach must rest on organizational commitment to **quality** which fosters a continuous process improvement culture

Software Engineering - A Layered Technology



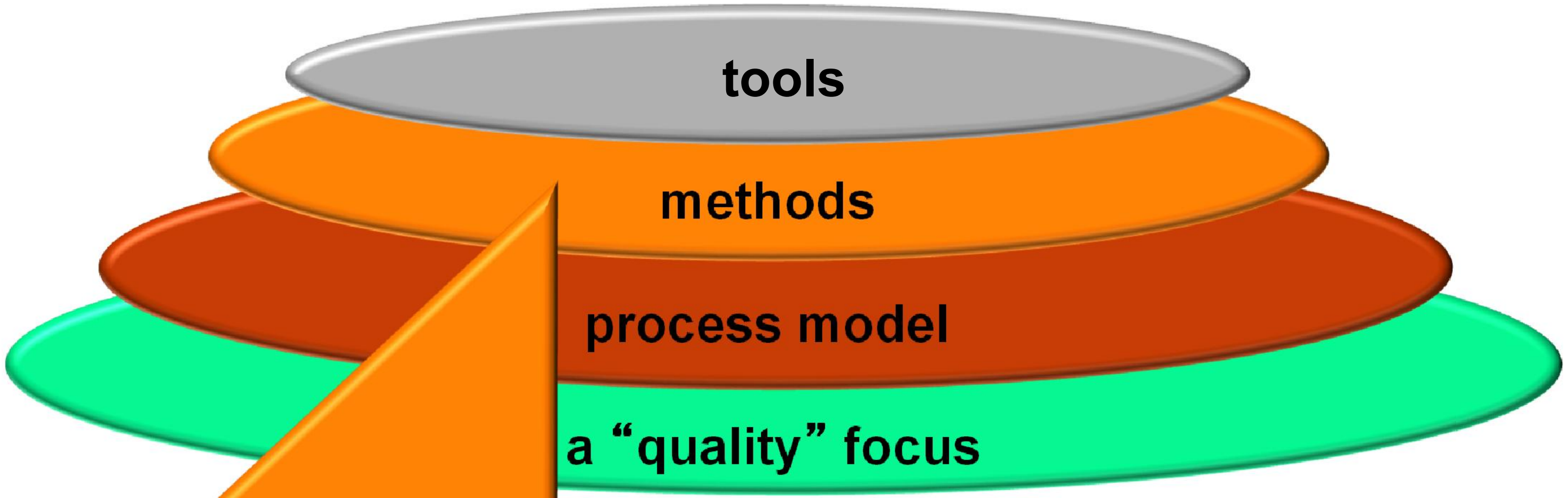
Process layer as the foundation defines a framework with activities for effective delivery of software engineering technology. Establish the context where products (model, data, report, and forms) are produced, milestone are established, quality is ensured and change is managed

Software Engineering - A Layered Technology



Method provides technical how-to‘s for building software. It encompasses many tasks including communication, requirement analysis, design modeling, program construction, testing and support

Software Engineering - A Layered Technology



Tools provide automated or semi-automated support for the process and methods

Software Engineering Process

Software Process

- A set of activities whose goal is the development or evolution of software

- Generic activities in all software processes are
  - Specification
    - What the system should do and its development constraints
  - Development
    - Production of the software system
  - Validation
    - Checking that the software is what the customer wants
  - Evolution
    - Changing the software in response to changing demands

What is a software process model?

- A simplified representation of a software process, presented from a specific perspective

- Examples of process perspectives are
  - Workflow perspective
    - Sequence of activities
  - Data-flow perspective
    - Information flow
  - Role/action perspective
    - Who does what

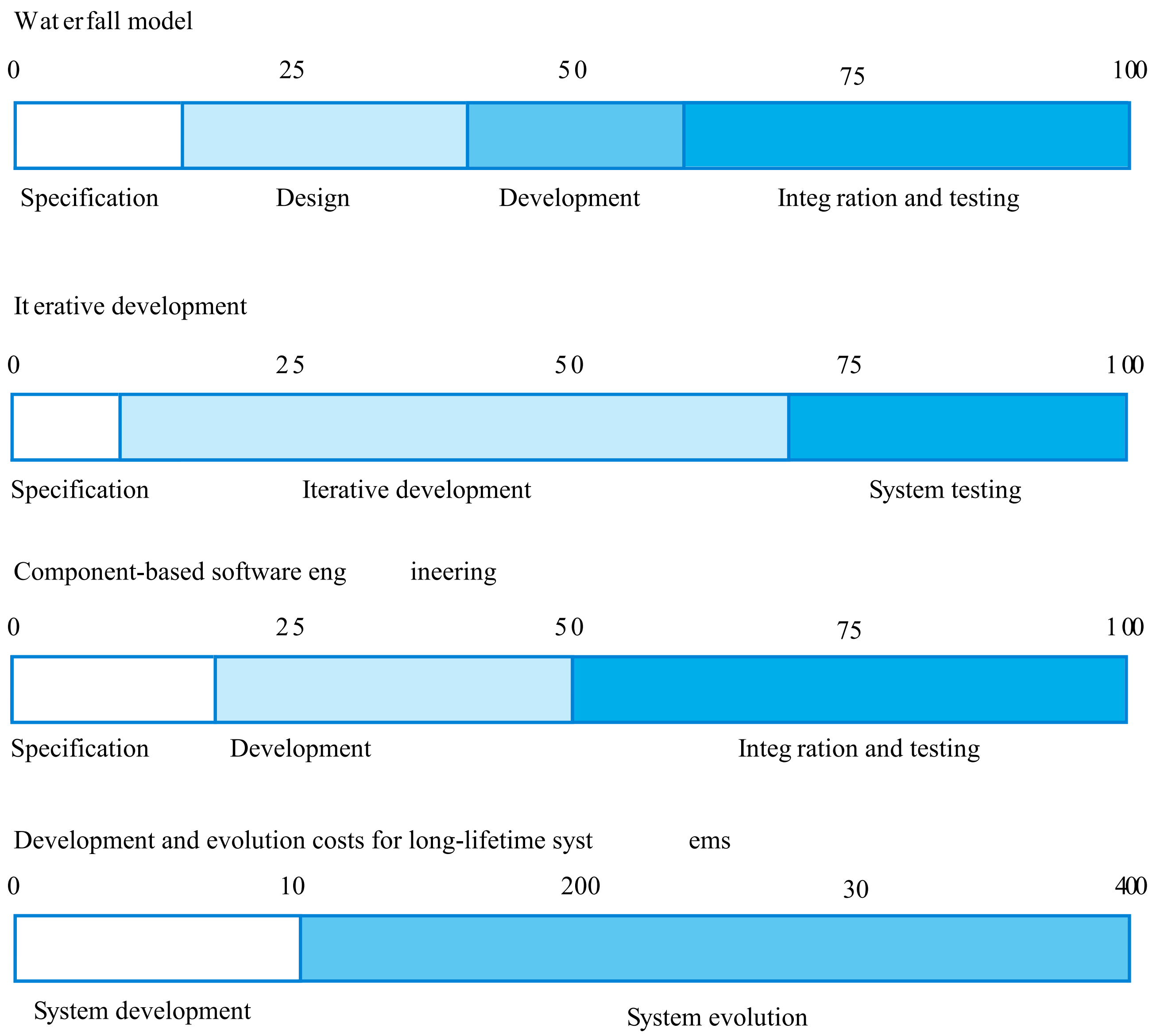
- Generic process models
  - Waterfall
  - Iterative development
  - Component-based software engineering

Software Engineering Costs

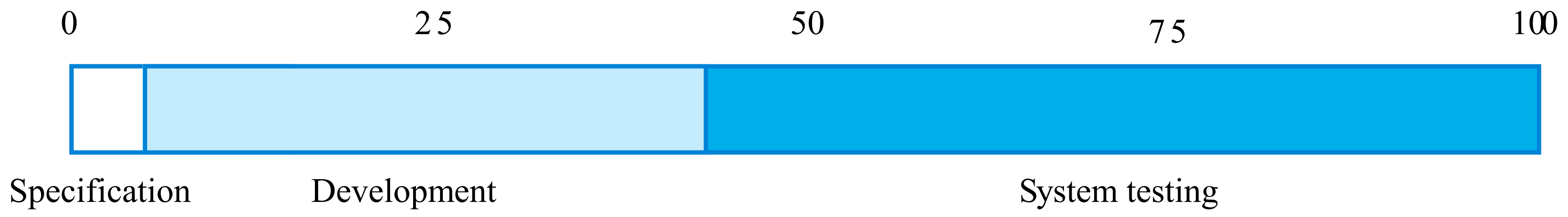
What are the Costs of Software Engineering?

- Roughly
  - 60% of costs are development costs
  - 40% are testing costs
  - **Note**: for custom software, evolution costs often exceed development costs.
- Costs vary depending on
  - The type of system being developed
  - The requirements of system attributes such as performance, system reliability, etc.
- Distribution of costs depends on the development model applied

Activity Cost Distribution



Product Developement Cost



- The majority od costs are spent for testing and correction of developement failures
- The developement costs cover about 35% of costs
- The specification costs cover about 5% of the costs

Software Engineering
-
Software Development

Dr.-Ing. Ali Diab

Outline

- Quote (Yogi Berra)
- Software Engineering Process
- Software Engineering Costs
- Software Developement Life Cycles (SDLC)
  - Bug & Fix Model
  - Waterfall Model
  - Rapid Prototyping Model
  - Incremental Model
  - Extreme Programming Model
  - Synchronize-and Stabilize Model
  - Spiral Model
  - Object-Oriented Life-Cycle Models

Quote

You've got to be very careful if you don't know where you're going, because you might end up there."

-Yogi Berra

Software Engineering Process

Software Process

- A set of activities whose goal is the development or evolution of software

- Generic activities in all software processes are
  - Specification
    - What the system should do and its development constraints
  - Development
    - Production of the software system
  - Validation
    - Checking that the software is what the customer wants
  - Evolution
    - Changing the software in response to changing demands

What is a software process model?

- A simplified representation of a software process, presented from a specific perspective

- Examples of process perspectives are
  - Workflow perspective
    - Sequence of activities
  - Data-flow perspective
    - Information flow
  - Role/action perspective
    - Who does what

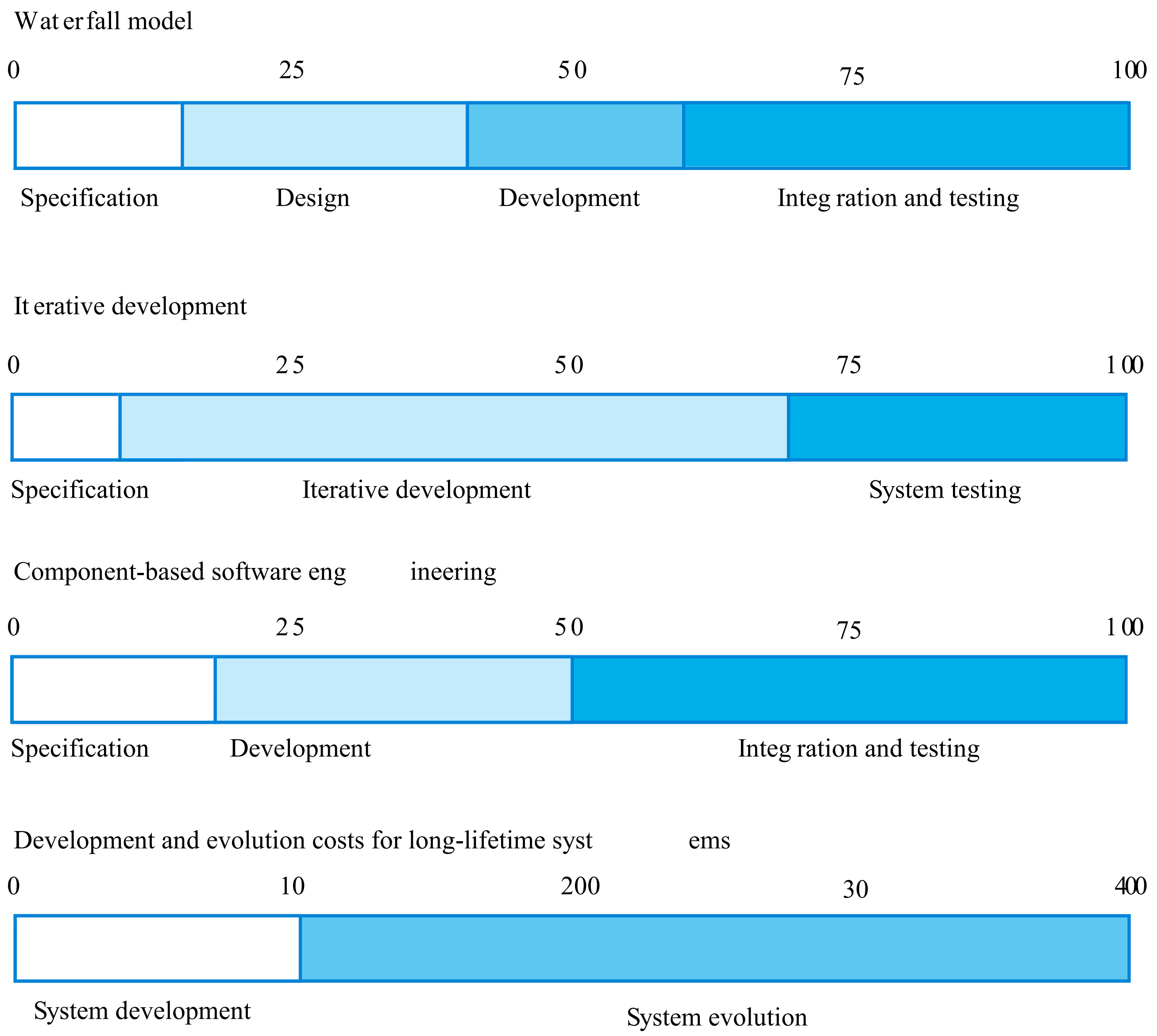
- Generic process models
  - Waterfall
  - Iterative development
  - Component-based software engineering

Software Engineering Costs

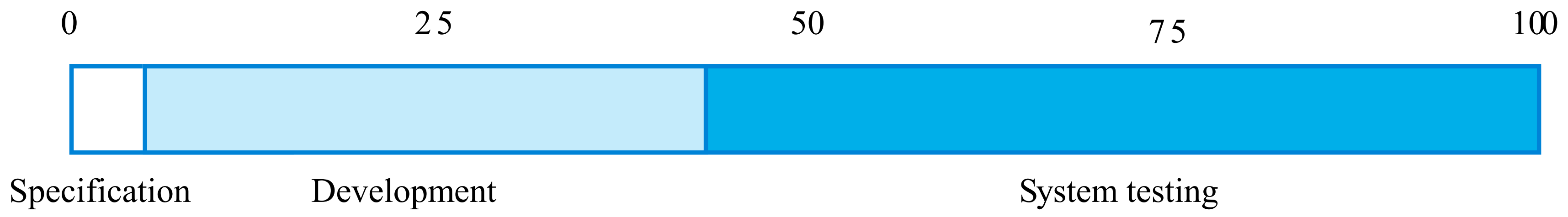
What are the Costs of Software Engineering?

- Roughly
  - 60% of costs are development costs
  - 40% are testing costs
  - **Note**: for custom software, evolution costs often exceed development costs.
- Costs vary depending on
  - The type of system being developed
  - The requirements of system attributes such as performance, system reliability, etc.
- Distribution of costs depends on the development model applied

Activity Cost Distribution



Product Developement Cost



- The majority od costs are spent for testing and correction of developement failures
- The developement costs cover about 35% of costs
- The specification costs cover about 5% of the costs

Software Developement Life Cycles (SDLC)

SDLC Model

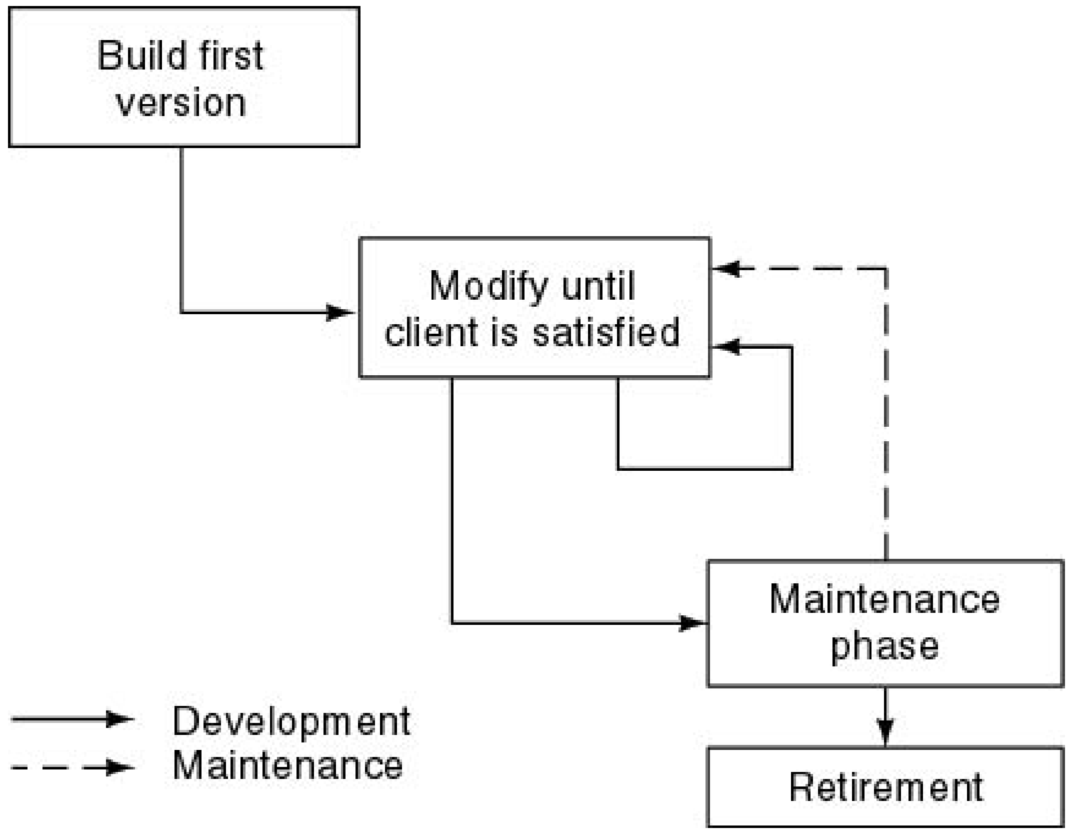
- A framework that describes the activities performed at each stage of a software development project
  - Bug & Fix Model
  - Waterfall Model
  - Rapid Prototyping Model
  - Incremental Model
  - Extreme Programming Model
  - Synchronize-and Stabilize Model
  - Spiral Model
  - Object-Oriented Life-Cycle Models
  - Dynamic Systems Development Method (DSDM)
  - Adaptive Model
  - Tailored Models
  - ...

Software Developement Life Cycles (SDLC)

Build & Fix Model

Build & Fix Model

- Product is constructed with no specifications



Difficencies

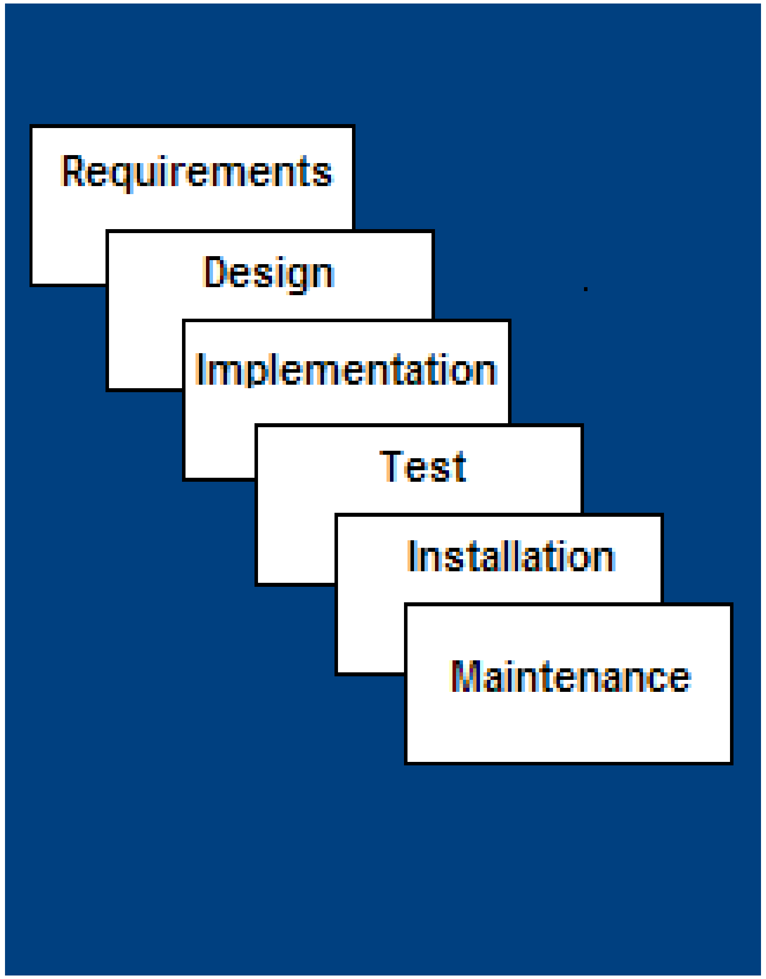
- Problems
  - No specifications
  - No design
- Totally unsatisfactory
- Need life-cycle model
  - “Game plan”
  - Phases
  - Milestones
- Disadvantages
  - Changes are made in the later phases of software development life cycle → **high cost**
  - Maintenance is difficult

Software Developement Life Cycles (SDLC)

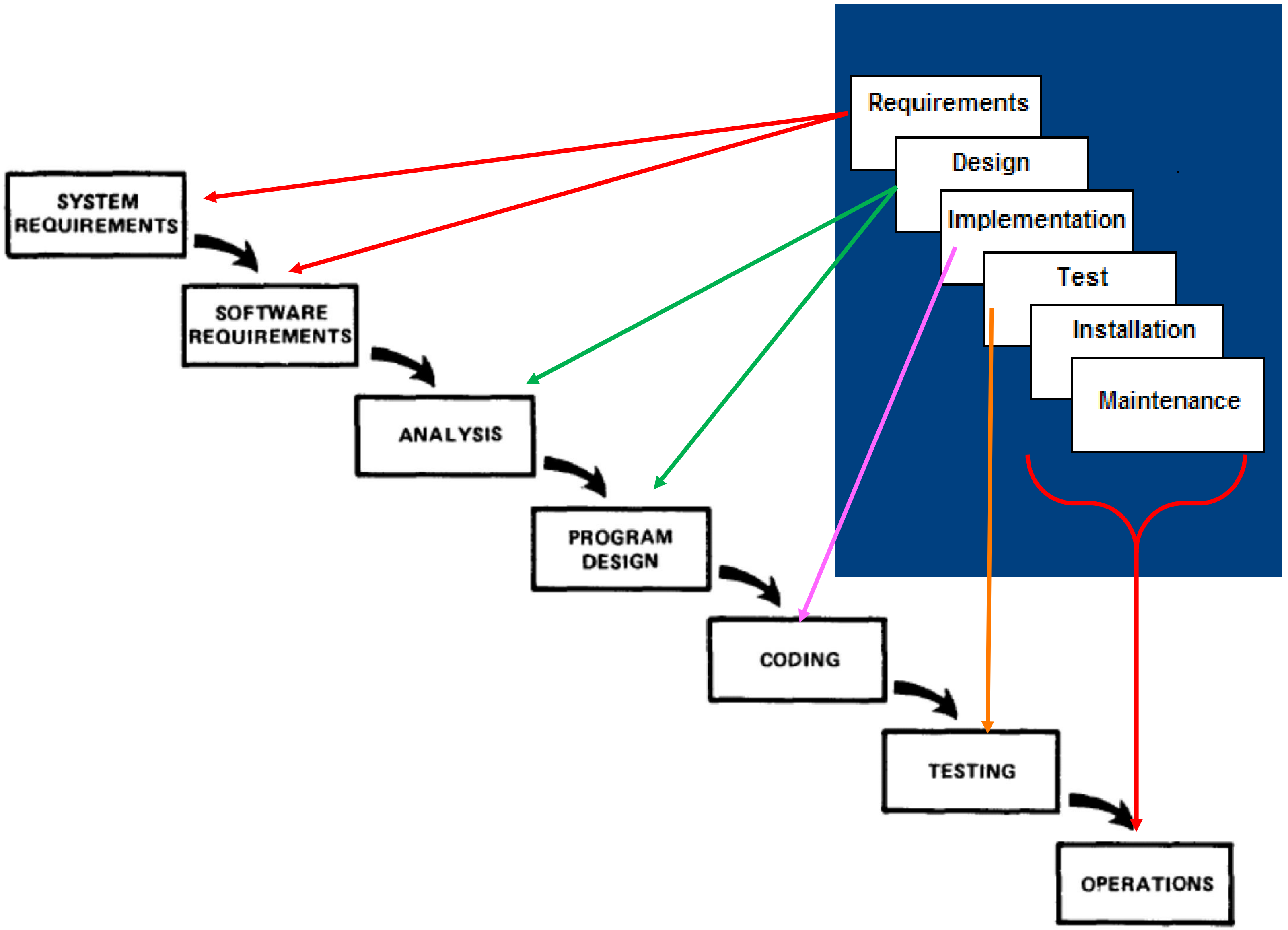
Waterfall Model

Waterfall Model

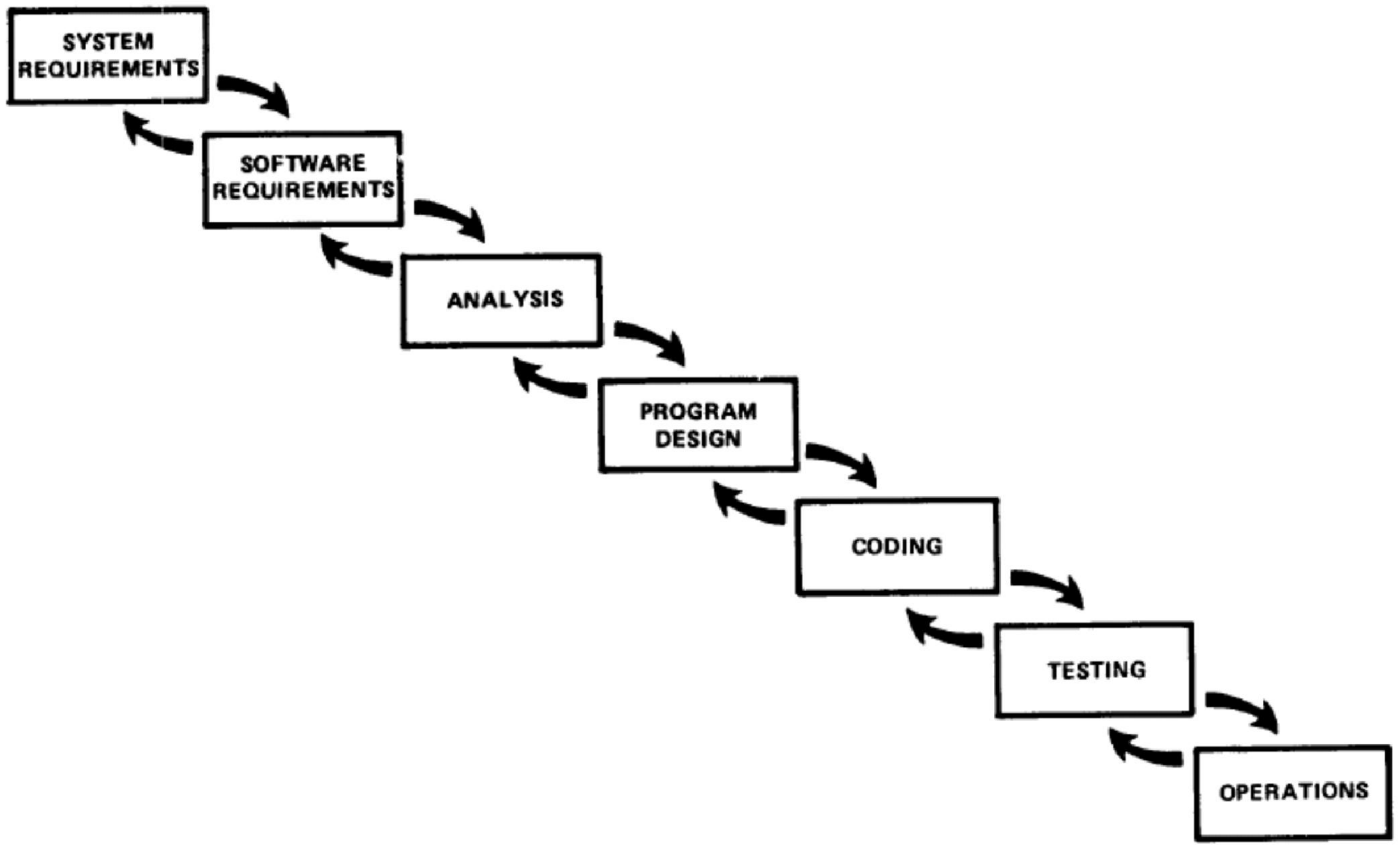
- **Requirements**, defines
  - Needed information
  - Functions & behavior
  - Performance & interfaces
- **Design**
  - Data structures
  - Software architecture
  - Interface representations
  - Algorithmic details
- **Implementation**
  - Source code & database
  - User documentation & testing



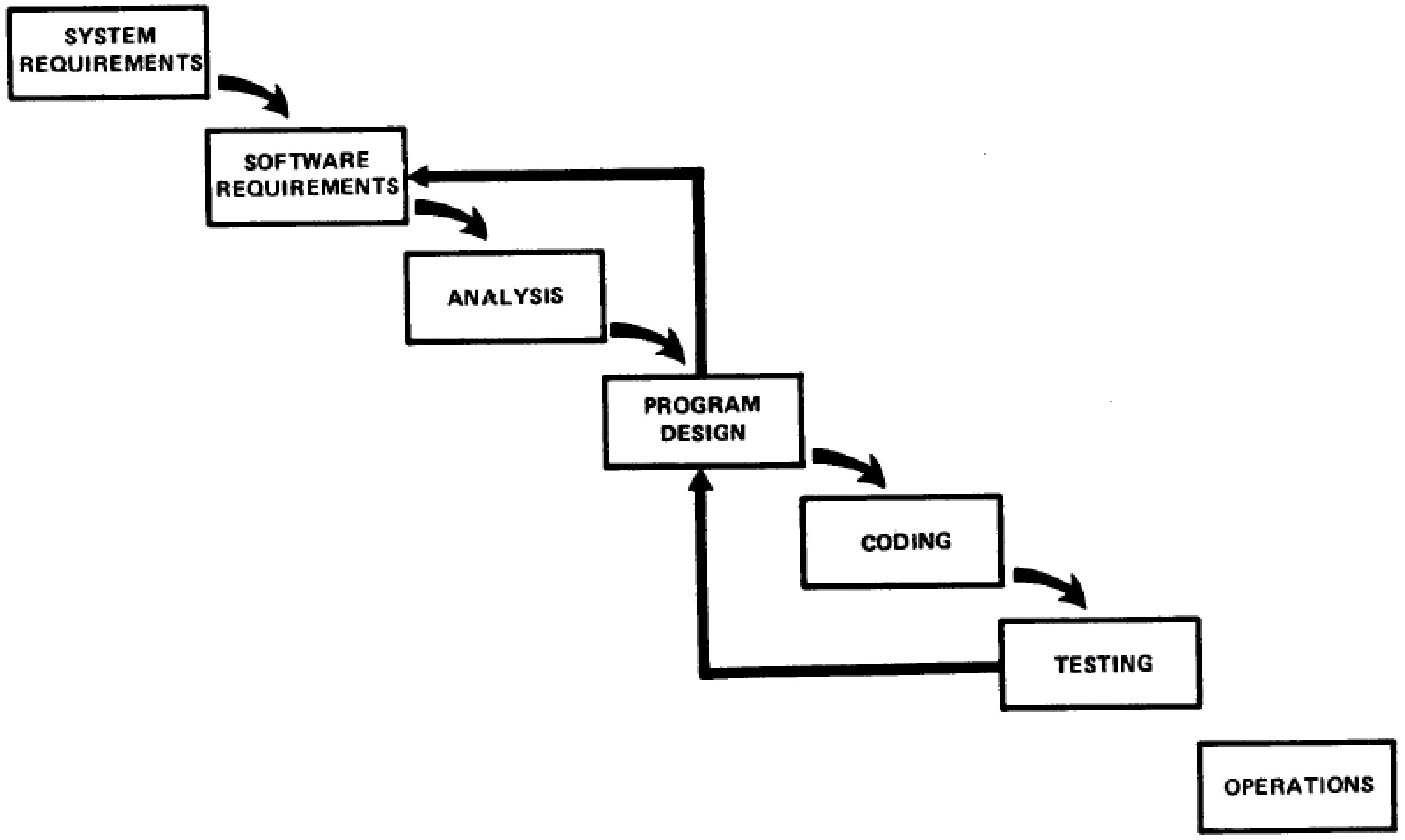
Waterfall Model



Waterfall Model



Waterfall Model

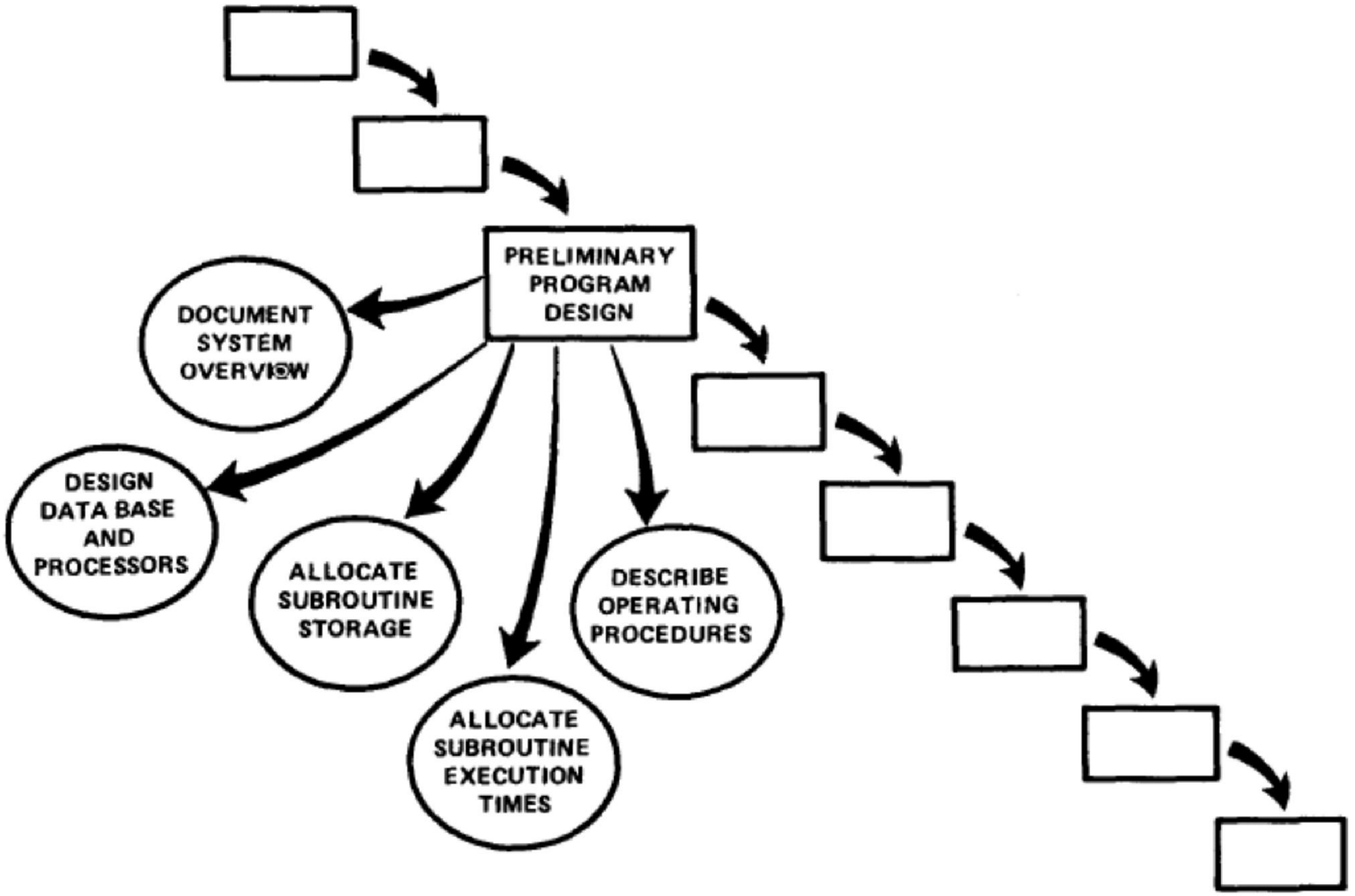


Development Risks Elimination

- Five additional features that must be added to this basic approach to eliminate most of the development risks
  - Step 1: Program design comes first
  - Step 2: Document the design
  - Step 3: Do it twice
  - Step 4: Plan, control and monitor testing
  - Step 5: Involve the customer

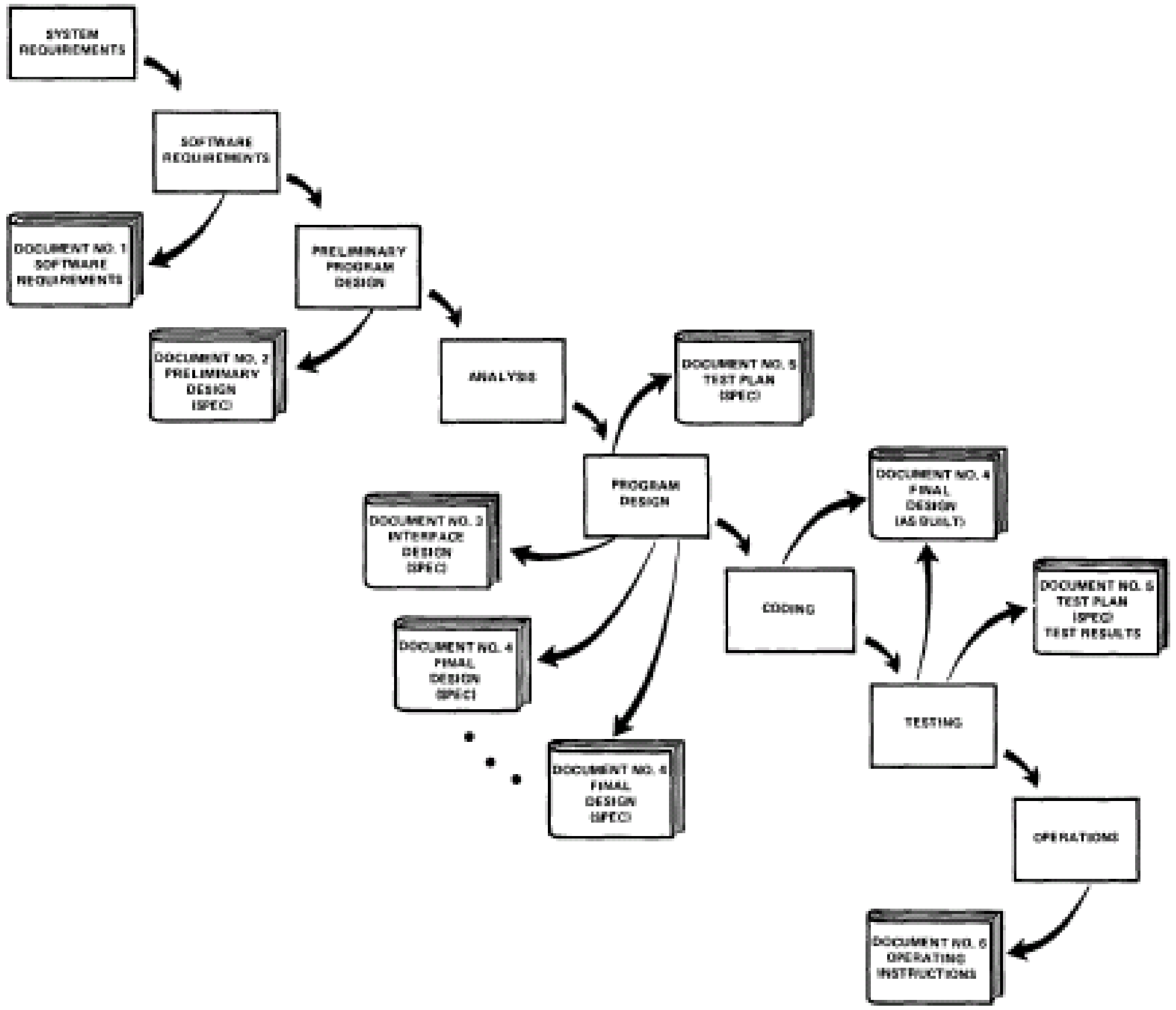
Development Risks Elimination

- Step1


PRELIMINARY PROGRAM DESIGN
DOCUMENT SYSTEM OVERVIEW
DESIGN DATA BASE AND PROCESSORS
ALLOCATE SUBROUTINE STORAGE
ALLOCATE SUBROUTINE EXECUTION TIMES
DESCRIBE OPERATING PROCEDURES

Development Risks Elimination

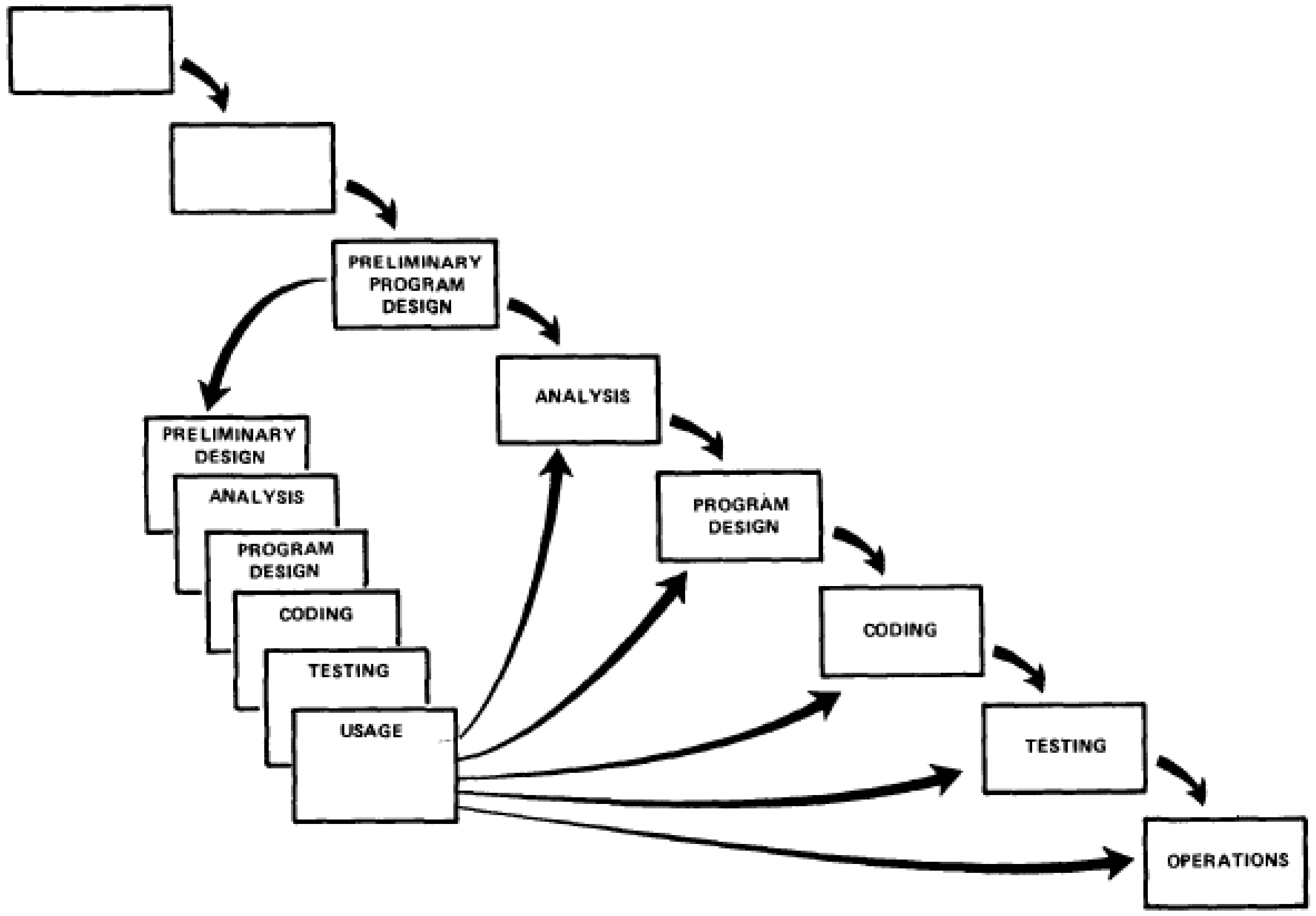
- Step 2



SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
DOCUMENT NO. 1 SOFTWARE REQUIREMENTS
PRELIMINARY PROGRAM DESIGN
DOCUMENT NO. 2 PRELIMINARY DESIGN (SPEC)
ANALYSIS
DOCUMENT NO. 5 TEST PLAN (SPEC)
PROGRAM DESIGN
DOCUMENT NO. 3 INTERFACE DESIGN (SPEC)
DOCUMENT NO. 4 FINAL DESIGN (SPEC)
DOCUMENT NO. 4 FINAL DESIGN (SPEC)
DOCUMENT NO. 4 FINAL DESIGN (AS BUILT)
CODING
DOCUMENT NO. 5 TEST PLAN (SPEC) TEST RESULTS
TESTING
OPERATIONS
DOCUMENT NO. 6 OPERATING INSTRUCTIONS

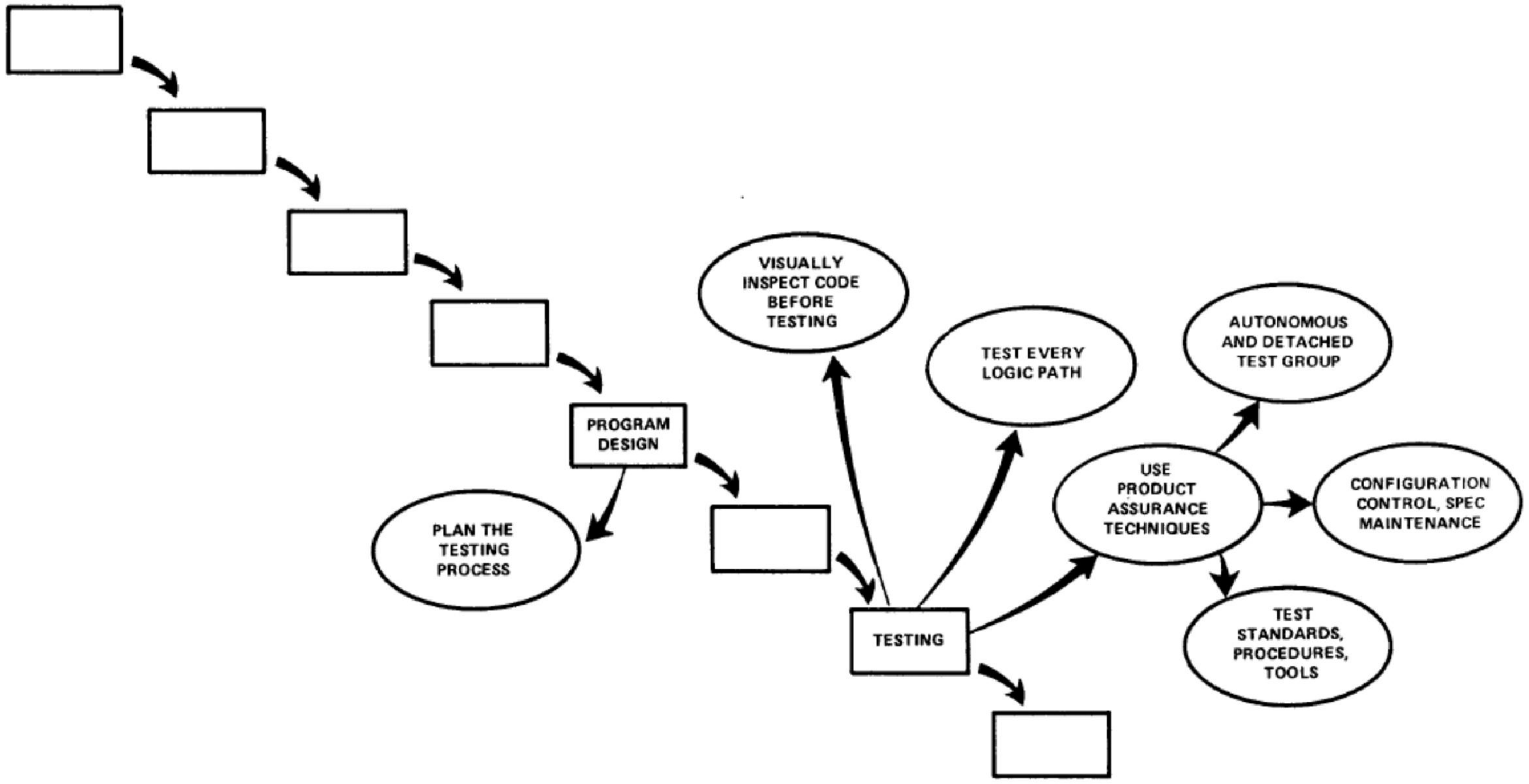
Development Risks Elimination

- Step 3



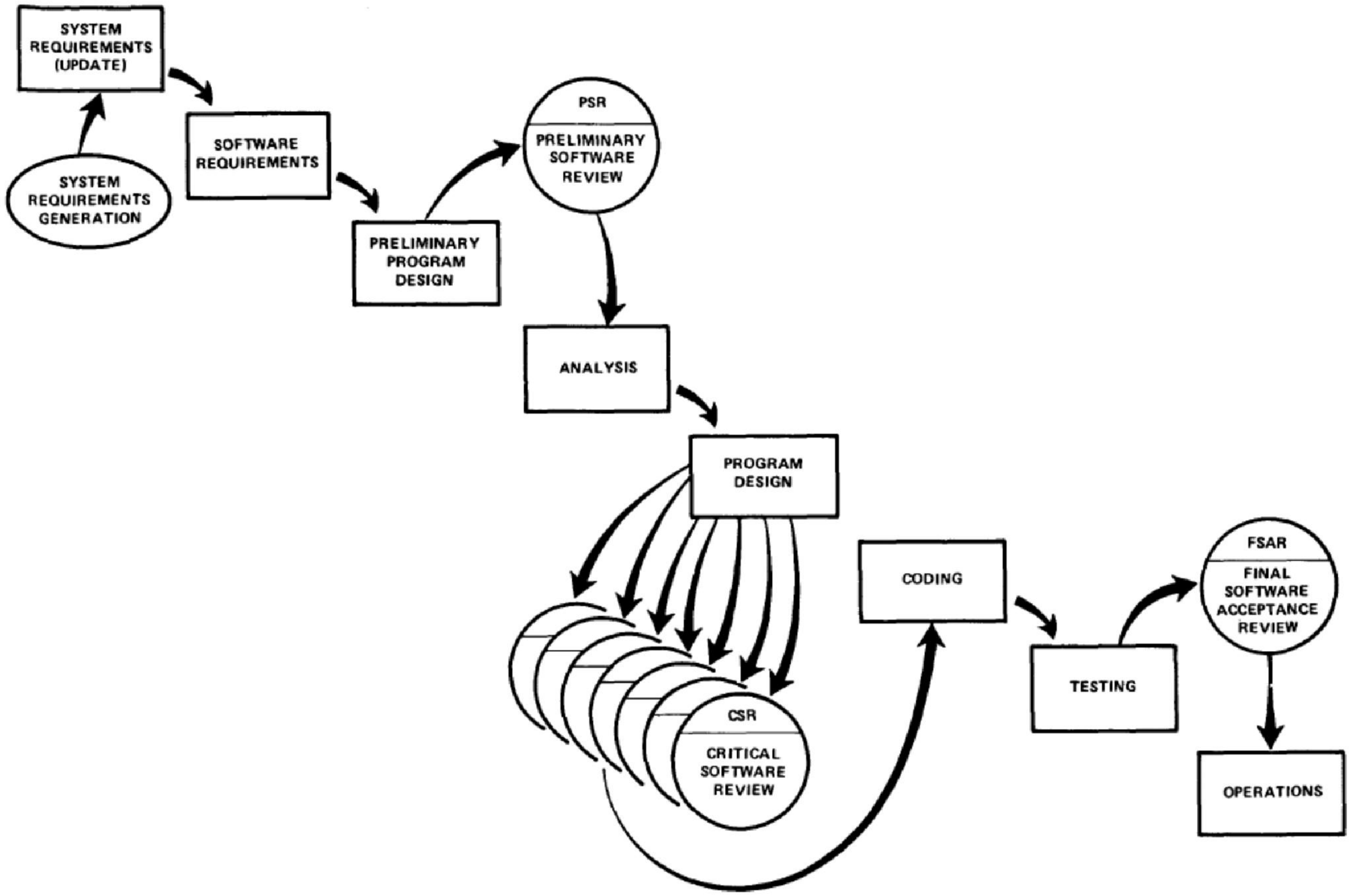
Development Risks Elimination

- Step 4


PROGRAM DESIGN
PLAN THE TESTING PROCESS
VISUALLY INSPECT CODE BEFORE TESTING
TEST EVERY LOGIC PATH
AUTONOMOUS AND DETACHED TEST GROUP
USE PRODUCT ASSURANCE TECHNIQUES
CONFIGURATION CONTROL, SPEC MAINTENANCE
TEST STANDARDS, PROCEDURES, TOOLS
TESTING

Development Risks Elimination

- Step 5



SYSTEM REQUIREMENTS (UPDATE)
SYSTEM REQUIREMENTS GENERATION
SOFTWARE REQUIREMENTS
PRELIMINARY PROGRAM DESIGN
PSR
PRELIMINARY SOFTWARE REVIEW
ANALYSIS
PROGRAM DESIGN
CSR
CRITICAL SOFTWARE REVIEW
CODING
TESTING
FSAR
FINAL SOFTWARE ACCEPTANCE REVIEW
OPERATIONS

Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements in a stable way
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

Deficiencies

- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

When to use the Waterfall Model

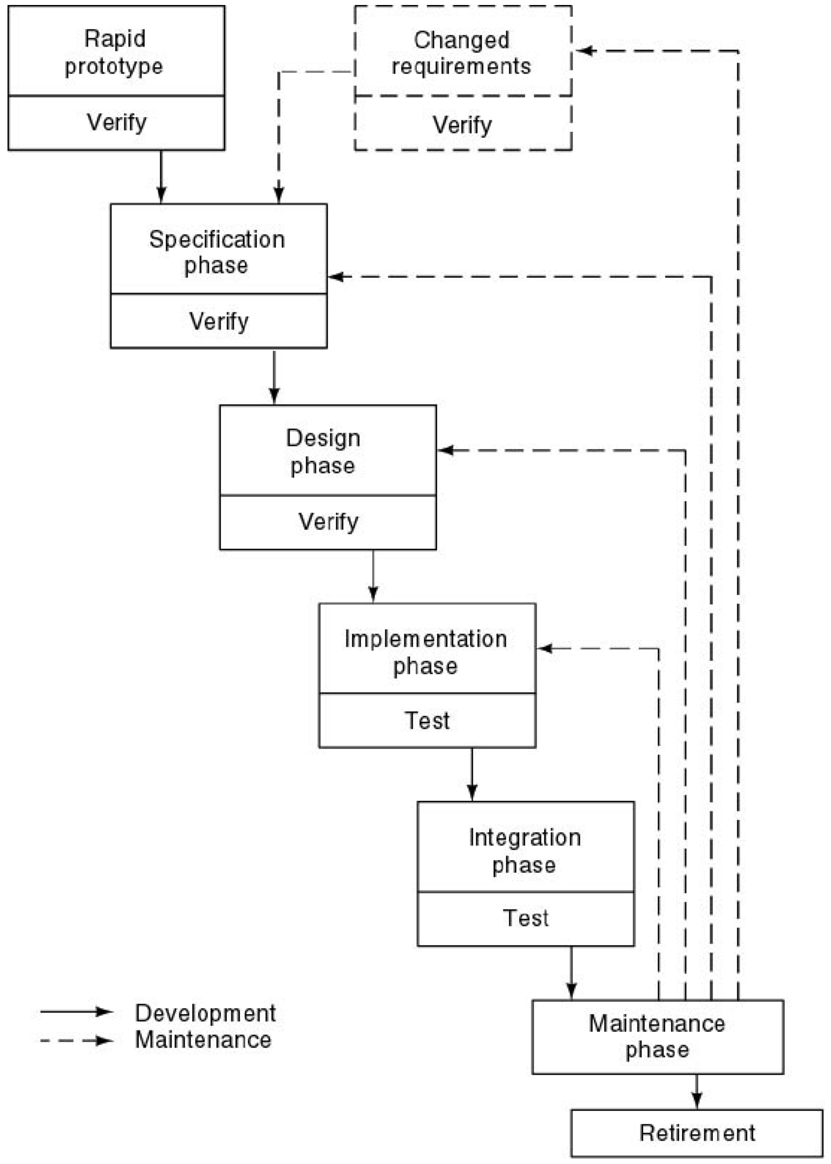
- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform
- Whether you should use it or not depends largely on
  - How well you believe you understand your customer's needs
  - How much volatility you expect in those needs as the project progresses

Software Developement Life Cycles (SDLC)

Rapid Prototyping Model

Rapid Prototyping Model

- Build a rapid prototype
- Clients and future users interact and experiment with it
- Once clients are satisfied, the developer can draw up specification document
- Feedback loops are not required in this model as the working prototype has been validated by the client→ it is reasonable to expect that the resulting specification document will be correct



The Prototyping

- Construct the prototype as rapidly as possible
  - The developers should develop the rapid prototype as rapid as possible to speed up the software development process
- Use of rapid prototype
  - Determine what the client’s real needs are
- Rapid   prototype implementation is discarded
  - Internal structure of rapid prototype is not relevant
  - Prototype is modified rapidly to reflect client need

Strengths & Difficincies

- Advantages
  - As compared to waterfall model
    - Development of the process is linear preceding from rapid prototype to delivered product
    - Feedback loops are less likely to be (needed) in this case

- Disadvantages
  - If users cannot be involved throughout the life cycle this model is not useful
  - Development time may not be reduced if reusable components are not available
  - Highly specialized and skilled developers are expected and such developers may not be available easily
  - Client expects changes to made as rapidly as the rapid prototype

Waterfall Model Vs. Rapid Prototyping Model

- Waterfall model
  - Many successes
  - Client needs
- Rapid prototyping model
  - Not proved
  - Has own problems
- Solution
  - Rapid prototyping for requirements phase
  - Waterfall for rest of life cycle

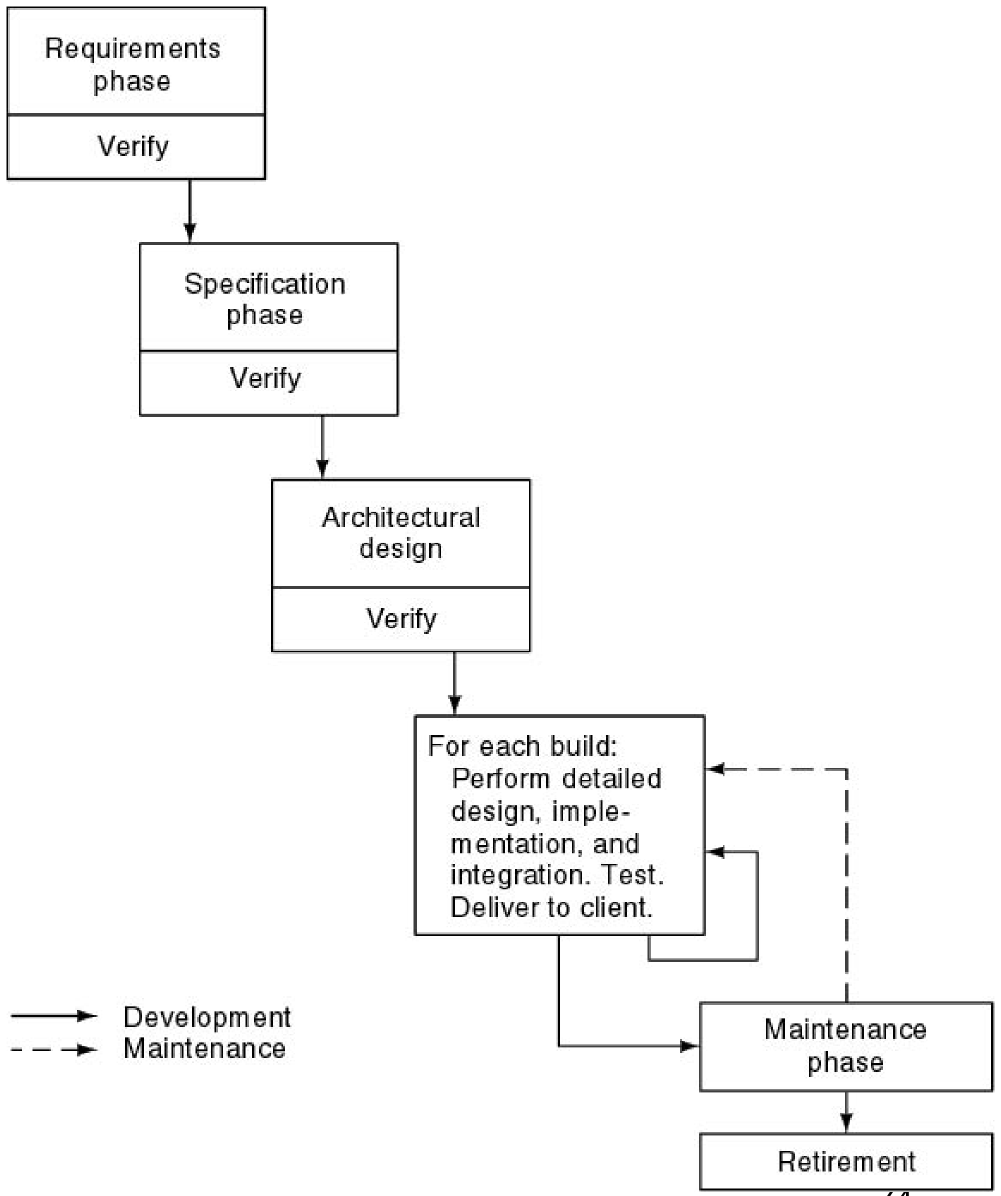
Software Developement Life Cycles (SDLC)

Incremental Model

Incremental Model

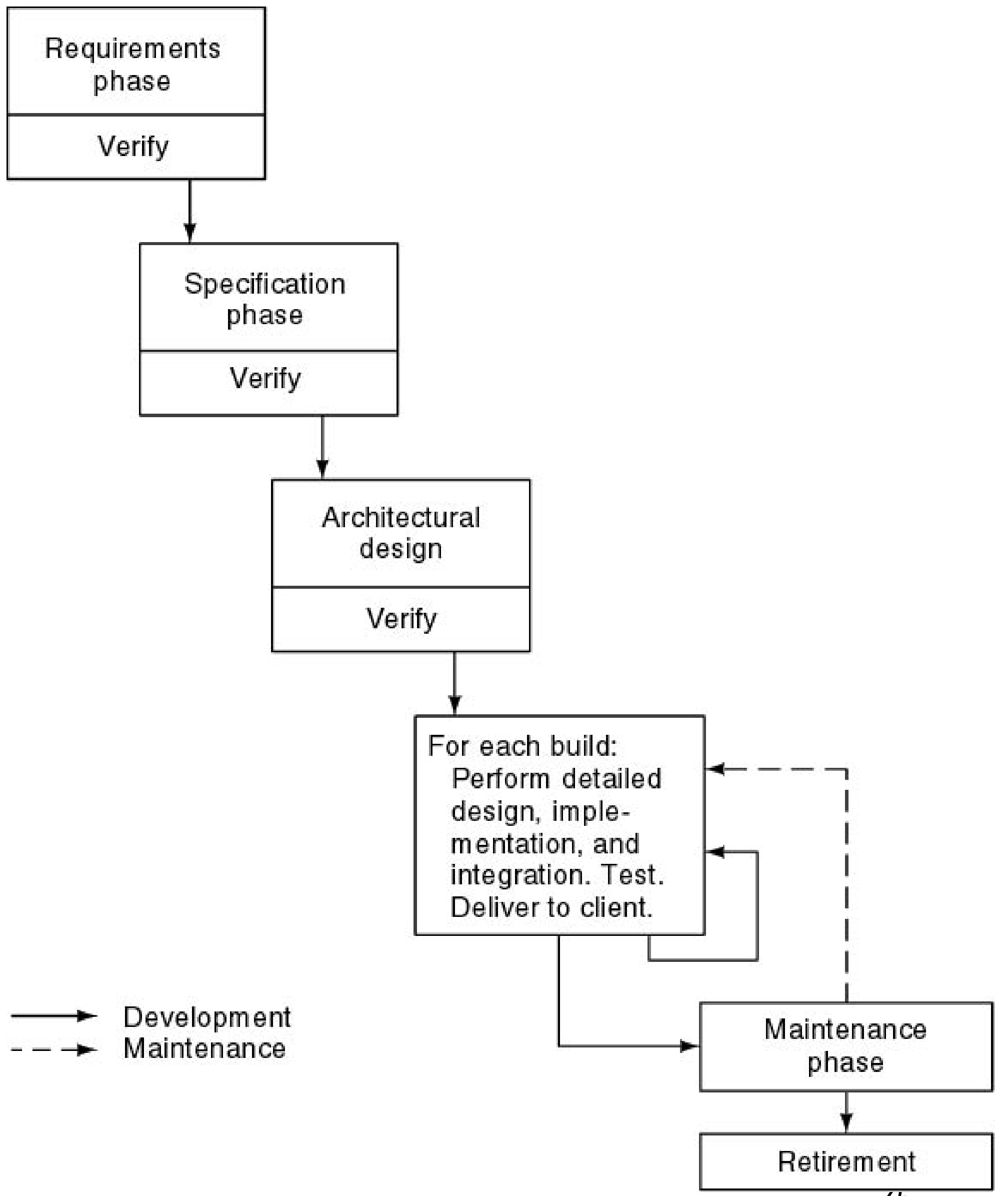
- The product is designed, implemented, integrated and tested as a series of incremental builds
  - A build consists of code pieces from various modules interacting to provide a specific functional capability

- At each stage, a new build is coded and then integrated and tested as a whole



Incremental Model

- Break up the target product into builds subject
  - Constraint that each build is integrated into existing software → the resulting product must be testable
- If a product has too many builds, then at each stage
  - Considerable time is spent in the integration testing of only a small amount of additional functionality
- If a product has too few build, then
  - The incremental model degenerated into build and fix model



Comparison to Previous Models

- In waterfall and rapid prototyping model
  - Deliver to client a complete product
  - There is project delivery date

- In Incremented Model
  - Deliver an operational quality product at each stage

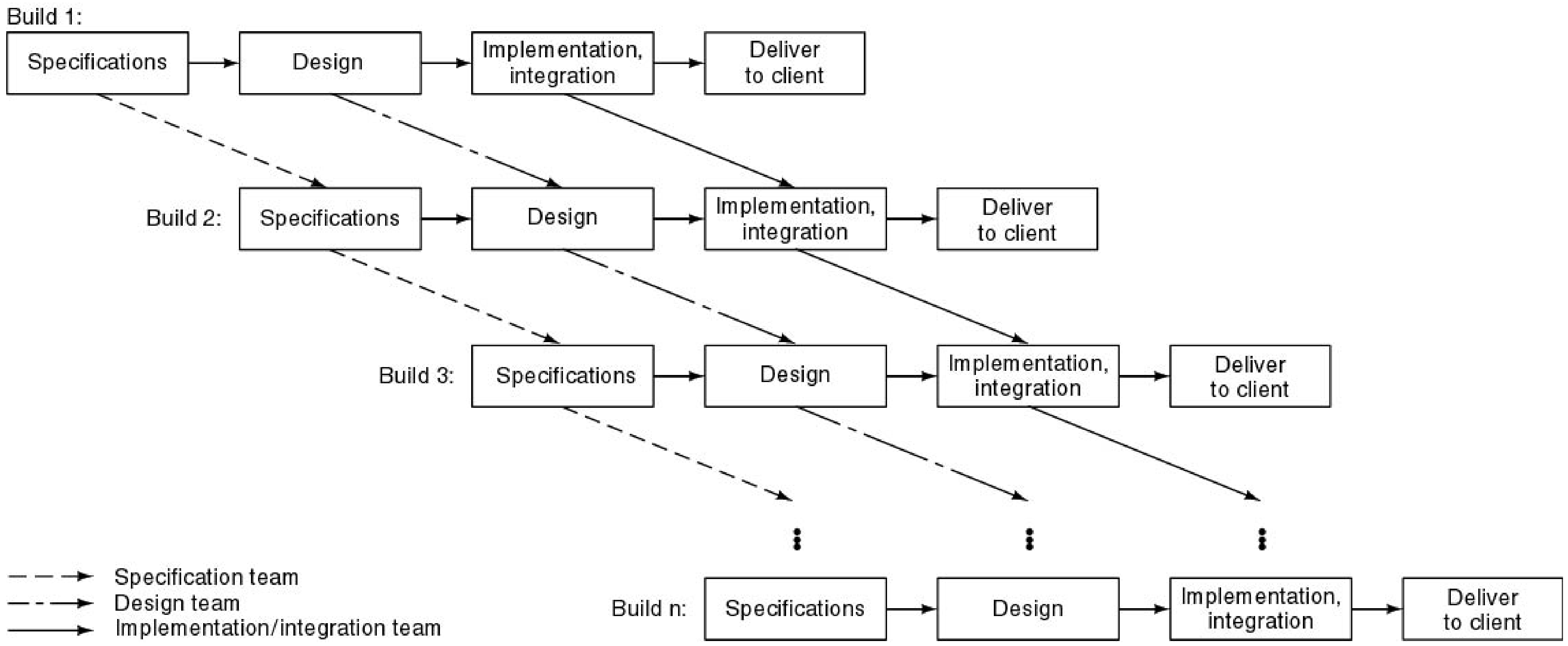
Strengths & Difficincies

- Advantages
  - Gradual introduction of the product via this model provides time for the client to adjust to the new product
  - Change and Adoptions are natural

- Disadvantages
  - Each additional build has to be incorporated into existing structure without destroying what has been made till date
  - Addition of new build should be simple and straightforward
  - Incremental model does not distinguish between developing a product and maintaining it
  - Begin with a design that support entire product
  - View product as a sequence of builds, each independent of next

Risk!!

- More risky concurrent version - pieces may not fit

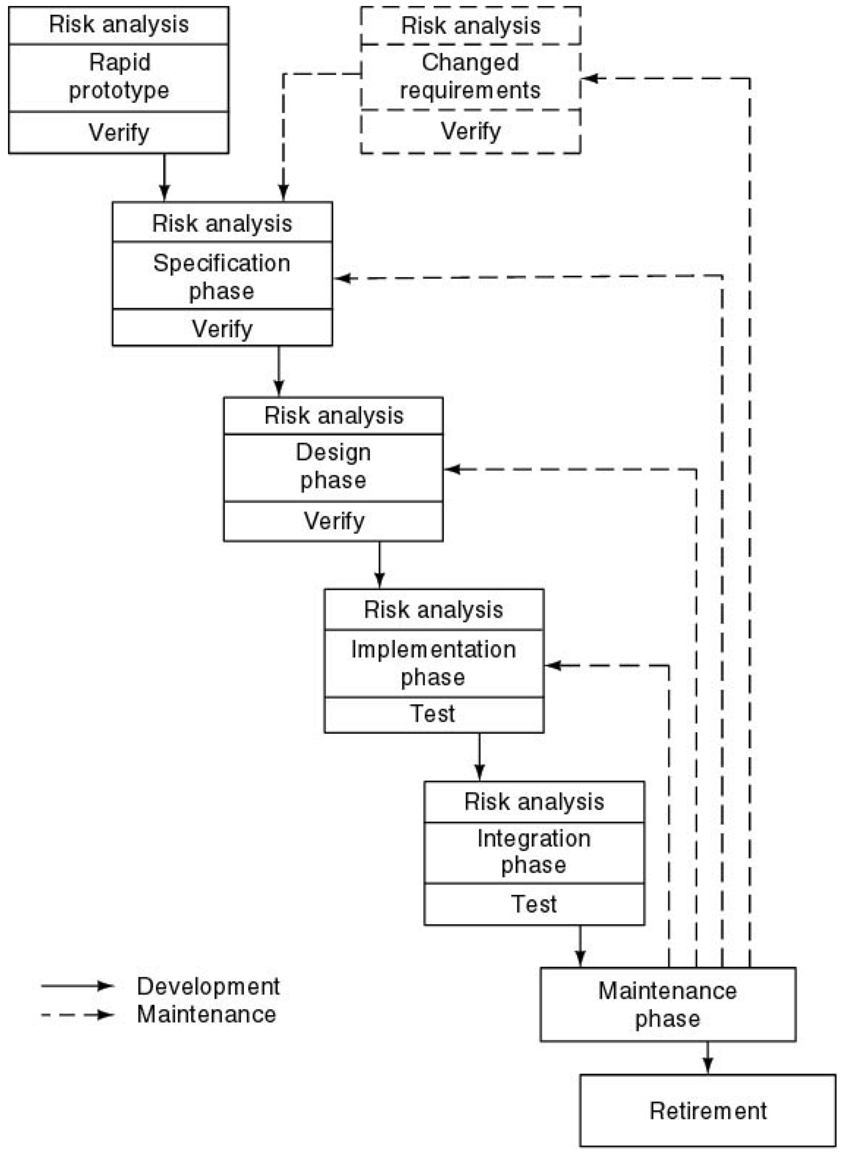


Software Developement Life Cycles (SDLC)

Spiral Model

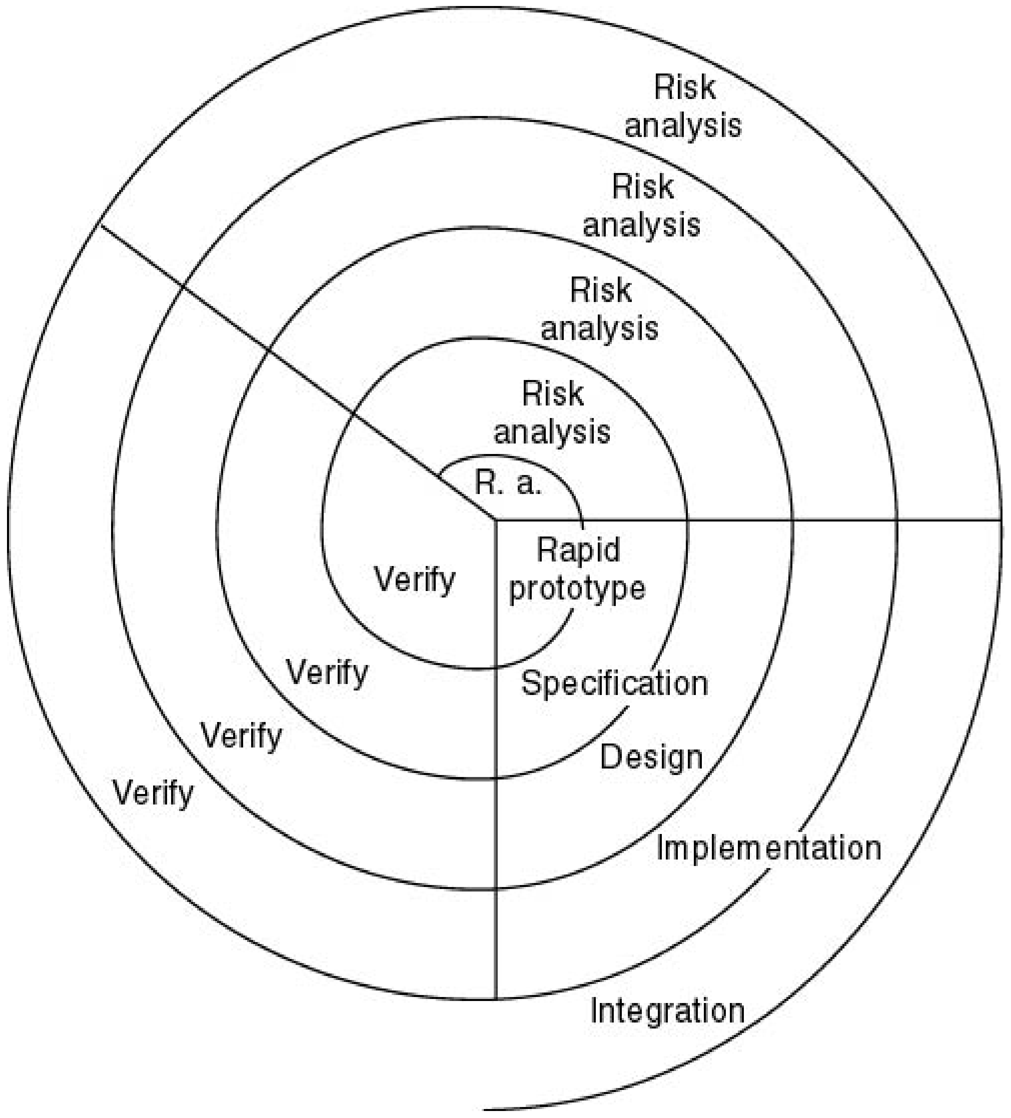
Spiral Model

- Simplified form
  - Waterfall model plus risk analysis
- Precede each phase by
  - Alternatives
  - Risk analysis
- Follow each phase by
  - Evaluation
  - Planning of next phase



Simplified Spiral Model

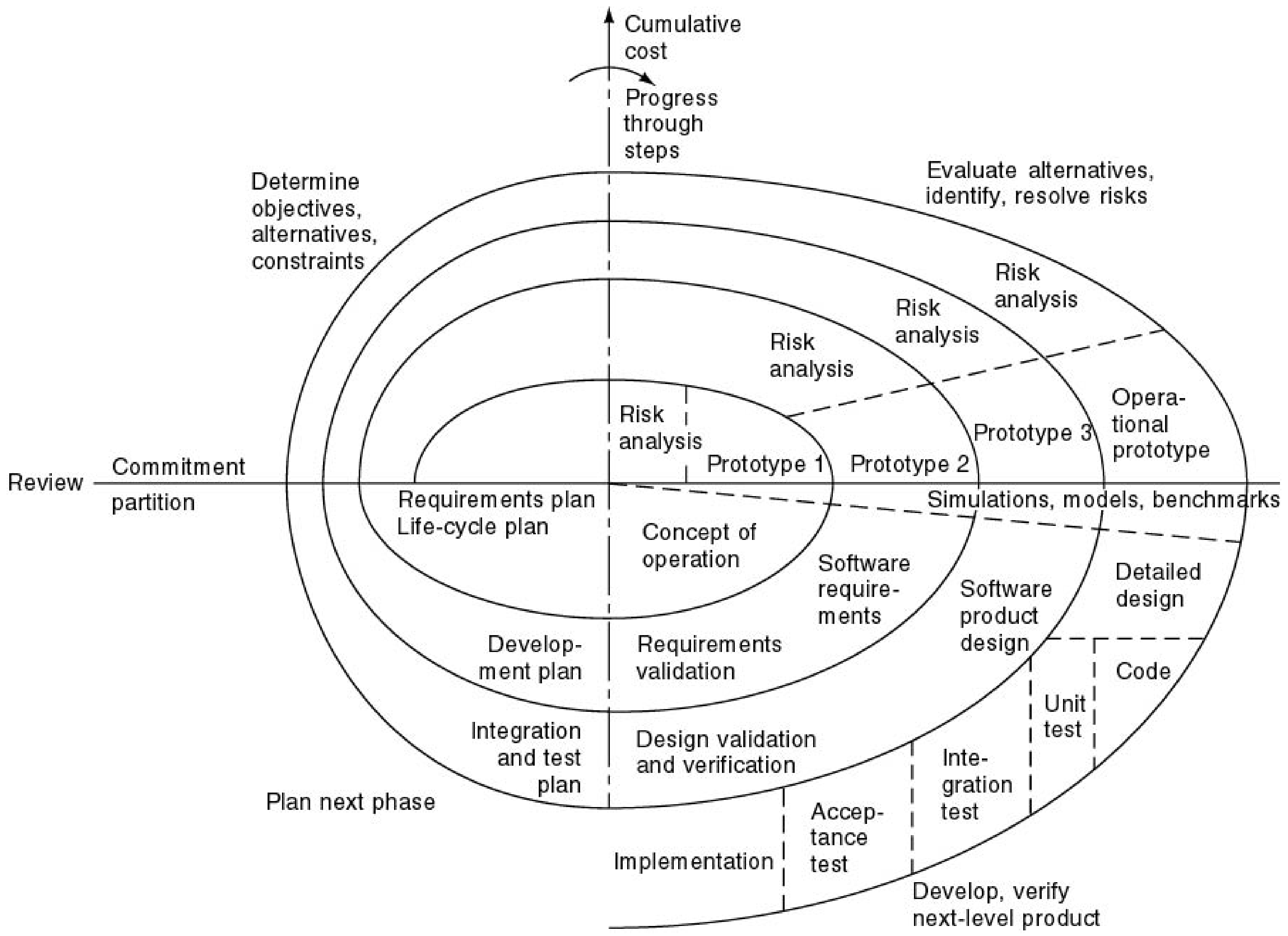
- If risks cannot be resolved, project is immediately terminated
- Prototypes can be effectively used to provide information about certain classes of risks



Full Spiral Model

- Represents cumulative cost to date and progress through the spiral
- Each cycle of the spiral corresponds to a phase
- Phase begins by determining objectives of that phase, alternatives for achieving those objectives, and constraints imposed on these alternatives
- Strategy is analyzed from view point of risk
- If all risks are successfully resolved development starts
- Then the results of the phase are evaluated

Full Spiral Model



Analysis of Spiral Model

- Strengths
  - Incorporation of software quality
  - Easy to judge how much to test as risks for too much and too low testing are analyzed
  - No distinction between development, maintenance i.e. maintenance is treated same way as development

- Weaknesses
  - For large-scale software only. In case of contract all risk analysis should be made before the contract is signed.
  - Skilled developers are required for analyzing and detecting potential risks
  - For internal (in-house) software only