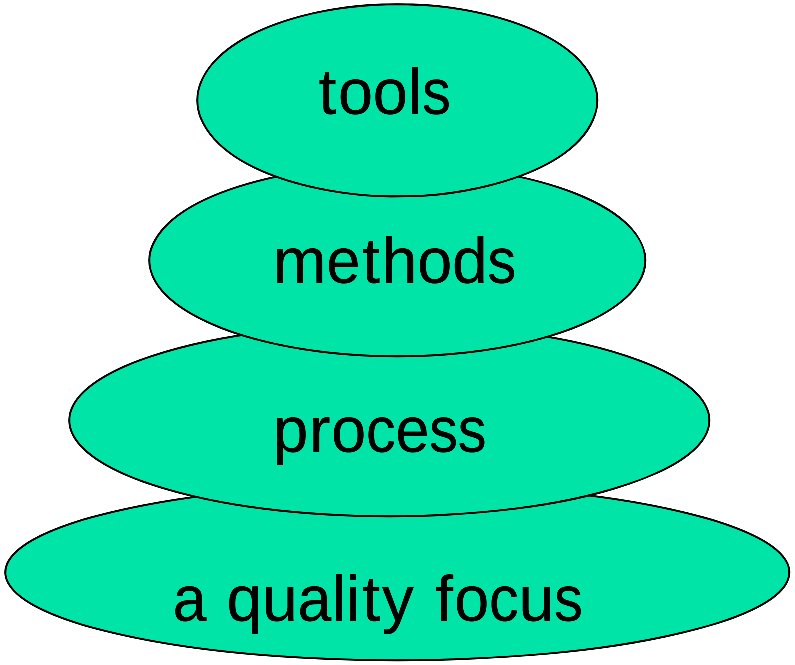
S.E. A Layered Technology



Layered Technology

- S.e is built on management’s commitment to quality
- Foundation – the process layer
- Process:
  - The glue that holds the technology layer together
  - Enables rational and timely development of software
  - Defines framework for key process areas (KPAs)
  - Involves a set of tools and techniques

Process Characteristics

- Prescribes all major process activities
- Use of resources subject to constraints
- Each process activity has entry and exit criteria
- Activities are organized in a sequence
- Guiding principles explain the goal of each activity

Layered Technology

- S.e approach must rest on an organization’s commitment to total quality
- TQM foster continuous process improvement which increasingly leads to more mature approaches to software engineering

Layered Technology

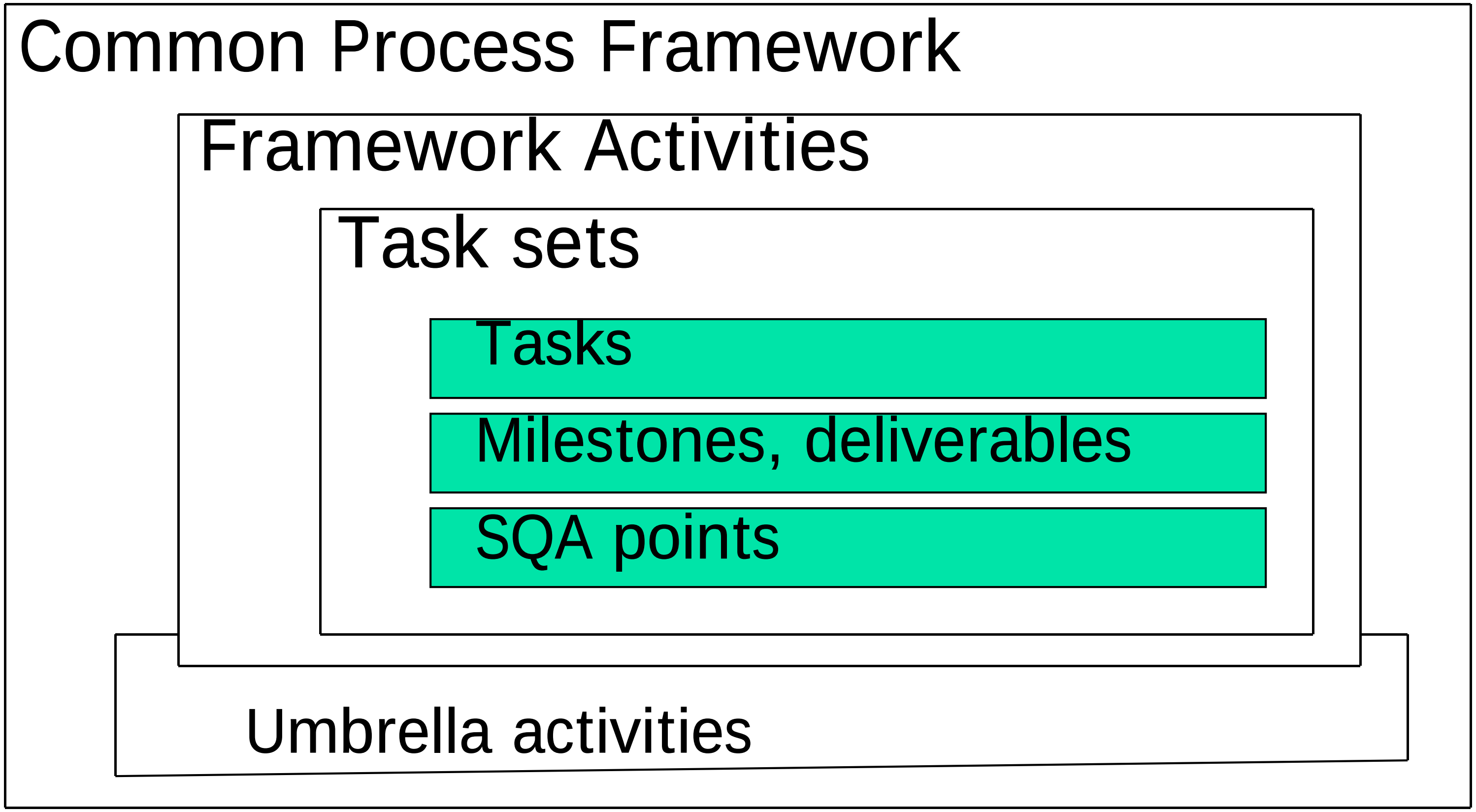
- Methods:
  - Provide technical how-to's for building software
  - Encompass a wide array of tasks, including
    - Requirements analysis
    - Design
    - Program construction
    - Testing
    - Support, etc

The Software Process

- Common framework process:

- Task sets:
  - Collection of s.e. work tasks
  - Project milestones
  - Software work products and deliverables
  - Quality assurance points

A Common Process Framework



Common Process Framework

Framework Activities

Task sets

Tasks

Milestones, deliverables

SQA points

Umbrella activities

KPA’s

- KPA’s – the basis for management control of software projects
  - Establish context for application of technical methods
  - Establish production of work products
  - Establish milestones
  - Measure quality
  - Manage change

Methods

- Technical “how to” in building software
- Include
  - requirements analysis,
  - design,
  - program construction,
  - testing,
  - maintenance
- Tools
  - Provide automated support for process and methods
  - CASE

Software Process

- Umbrella Activities
  - SQA
  - SCM
  - Measurement
  - Independent of framework activity
- Recent emphasis is on “Process Maturity”

CMM

- Improvement of development process is major challenge for organizations engaged in software production
- CMM is a popular method for process assessment and improvement
- Specified by SEI:
  - Originally used by US DOD to assess IT capabilities of organizations bidding for defense contracts
  - Now widely used by IT industry worldwide

The CMM

- Is based on knowledge acquired from software process assessment
- Is a model which provides organizations with more effective guidance for establishing process improvement programs
- Is the foundation for systematically building a set of tools including a maturity questionnaire, which are useful in software process improvement

CMM

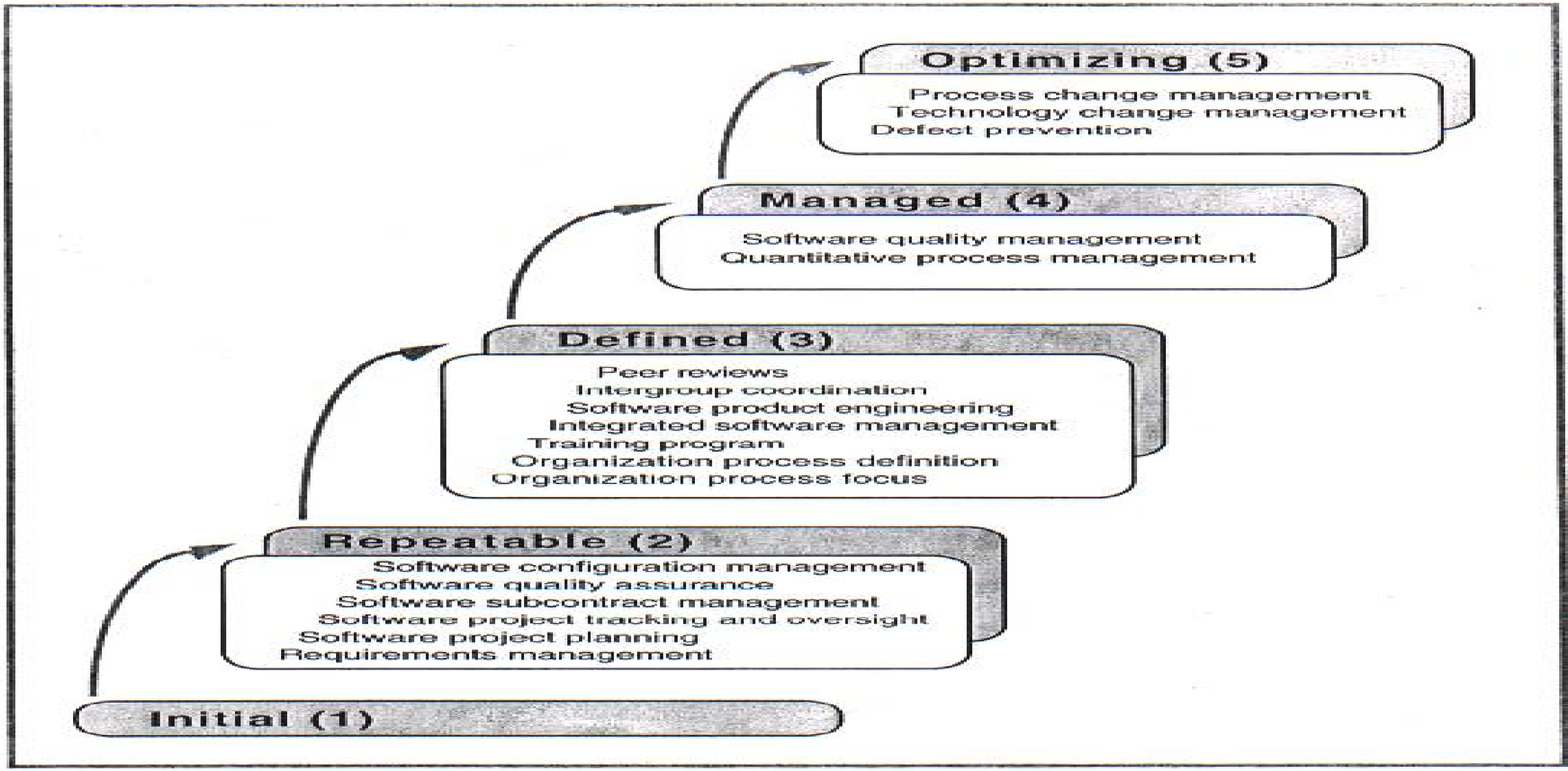
- Essentially a questionnaire an org fills out

- The SW-CMM is a framework that describes the elements of an effective software process

SW-CMM

- Each KPA is organized by common features that address the implementation environment
    - Each common feature contains key practices that describe activities which, when performed collectively, achieve a set of goals for achieving process maturity

- Each maturity level has its own goals which can be achieved by using the KPAs for that level



Optimizing (5)

Process change management
Technology change management
Defect prevention

Managed (4)

Software quality management
Quantitative process management

Defined (3)

Peer reviews
Intergroup coordination
Software product engineering
Integrated software management
Training program
Organization process definition
Organization process focus

Repeatable (2)

Software configuration management
Software quality assurance
Software subcontract management
Software project tracking and oversight
Software project planning
Requirements management

Initial (1)

Figure 1 The Key Process Areas by Maturity Level

KPAs Common Features

- Commitment to Perform
  - Describes organization's to ensure the process is established and will endure
- Ability to Perform
  - Describes the preconditions that must exist in the project or organization to implement the software process completely

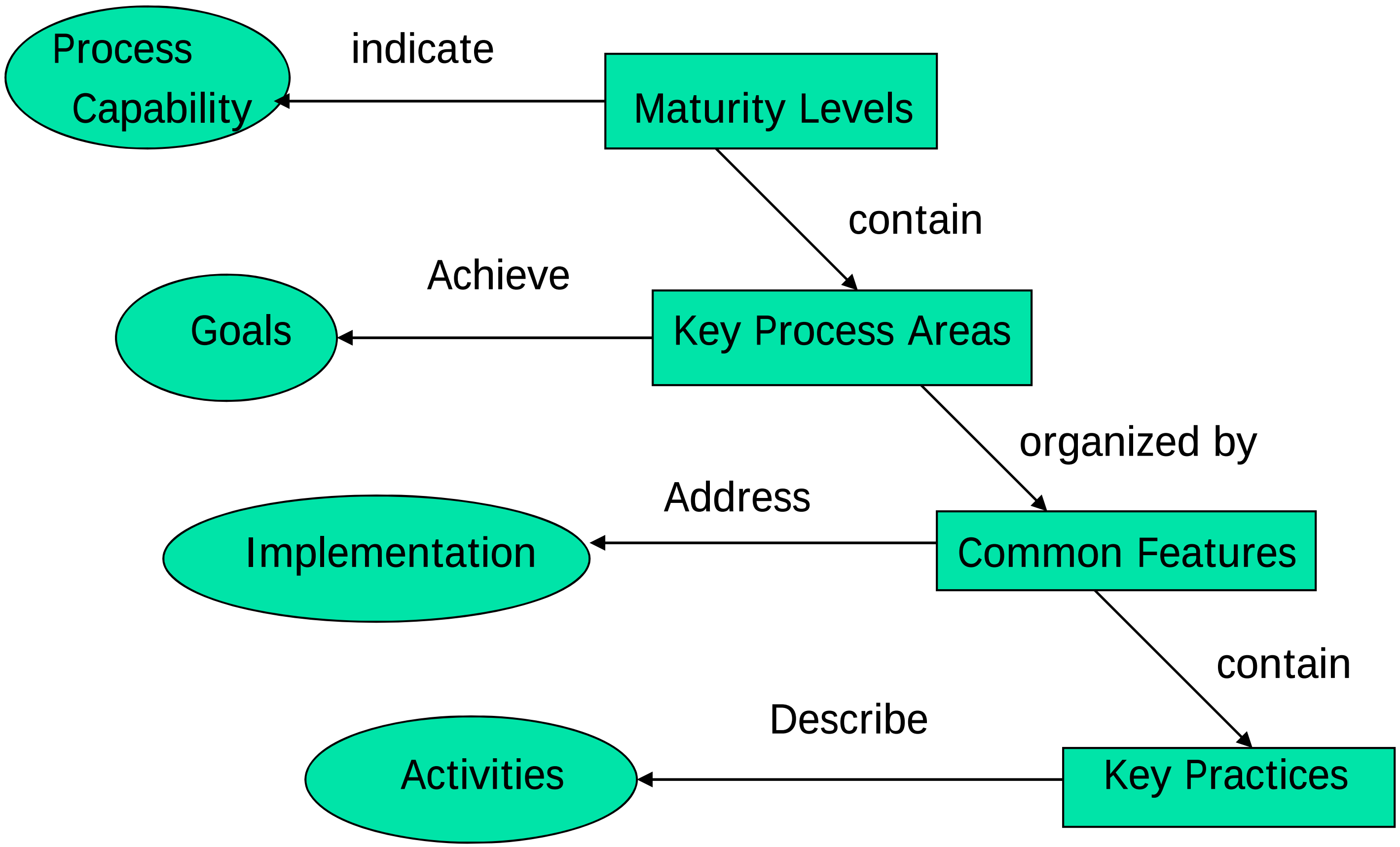
KPAs Common Features

- Activities Performed
  - Describes the roles and procedures necessary to implement a kpa
- Measurement and Analysis
  - Describes the need to measure the process and analyze the measurements
  - Typically includes examples of measurements that could be taken to determine the status and effectiveness of the activities performed

KPAs Common Features

- Verifying Implementation
  - Describes steps to ensure activities are performed in compliance with the process that has been established
  - Typically encompasses reviews and audits by management and SQA

Software CMM



CMM Grading Scheme

- Five-point scheme
  - Determines compliance with CMM that defines activities required at different levels of process maturity
  - Provides measure of global effectiveness of the company’s s.e. practices

CMM Levels

1 Initial: Ad hoc process. Success depends on individual effort

2. Repeatable: Basic management processes established to track cost, schedule, functionality. Necessary process discipline is in place to repeat earlier successes on similar projects

3. Defined: process for management and engineering activities is documented, standardized, and integrated into an organization-wide software process

4. Managed: Detailed measures are collected on software and product quality

5. Optimization: Continuous process improvement: quantitative feedback from the process and from testing new ideas and technologies

Other Process Improvement Models

- Besides CMM...
- ISO 9000 series of quality control
- ISO standards apply to quality management and the process to produce a quality product
- Standards are generic –

ISO 9000 Standard Premise

- If the process is right, the process outcome (product or service) will also be right
- The Objective of Quality Management is to produce quality products by building quality into the products rather than testing quality into the product
- ISO standards do not enforce or specify process

ISO Standards

- Provide models of what must be accomplished, not how activities must be performed
- Organization seeking ISO certification must:
  - say *wha*t it does
  - do what it says
  - demonstrate what it has done

ISO Standards

- Litmus test for an ISO certified organization:
  - It should be able to make a quality product or provide quality service even if its entire workforce were replaced...
    - Hence the organization must document and record all its activities
    - Written procedures must be defined for each activity, including what to do when things go wrong or customers complain

ISO Standards

- As with CMM, ISO certification can only be granted after an ***on-site audit*** by an ISO registrar
- Audits are repeated at regular intervals
- Organizations are forced into the scheme through

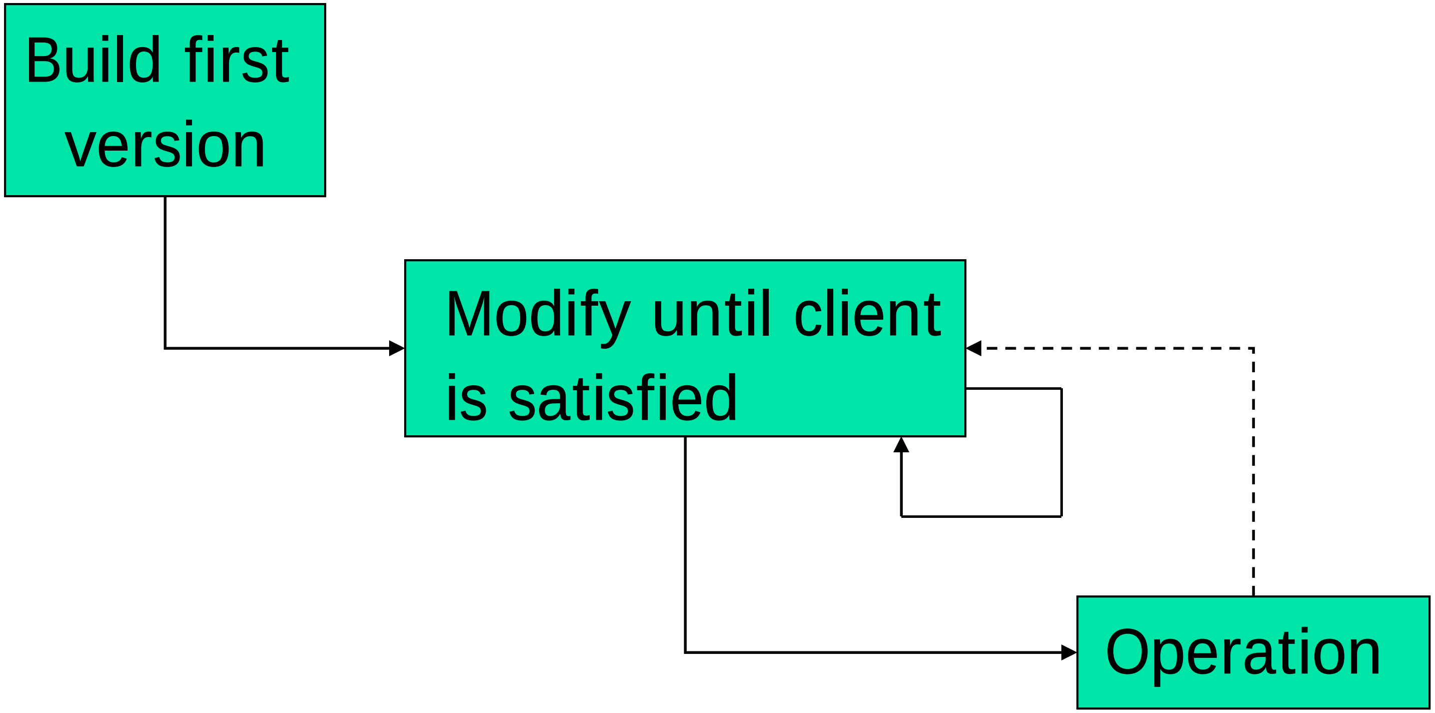
Level 2 Questionnaire, Example

1. Is there a successful data warehouse implementation within your company?
2. Is the successful data warehouse sustainable?
3. Do you have a meta data repository that, at a minimum, contains and manages data warehousing related meta data?
4. Has the successful data warehouse implemented development standards?
5. Does the enterprise lack centralized procedures/standards
6. Is your company spending extreme amounts on data warehousing (30 – 60% of IT budget)?
7. Does your company have multiple, large (8 to 15 staff members) data warehousing departments?

Paradigms

- Without planned life-cycle (a.k.a. process) model – the developer will work aimlessly
- A game plan
- Models:
  - Build-and-Fix
    - Build and deliver entire product
    - Client identifies desired changes
    - Satisfied client? → product in operation

Build-and-Fix Model



Build first version

Modify until client is satisfied

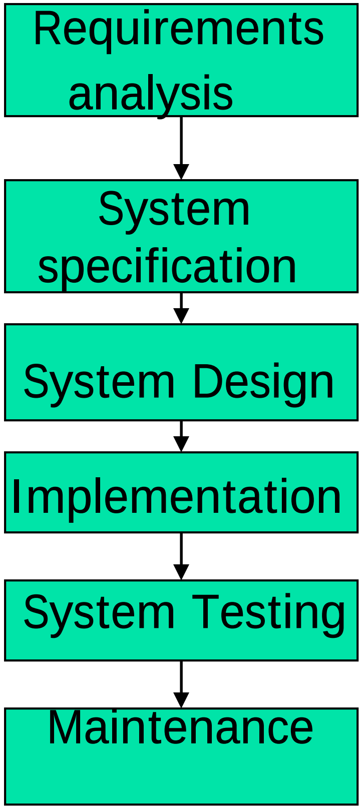
Operation

B-&-F Problems

- High cost of changes
- N0 specs, no planning, no design →
  - No coherent/cohesive overall structure
- Maintenance nightmare
- Not cost effective over life of product
- Correct time to use?
- Small with no possibility of future maintenance

Waterfall Model (Traditional)

- The software Development Process



Requirements analysis

System specification

System Design

Implementation

System Testing

Maintenance

Software Life Cycle

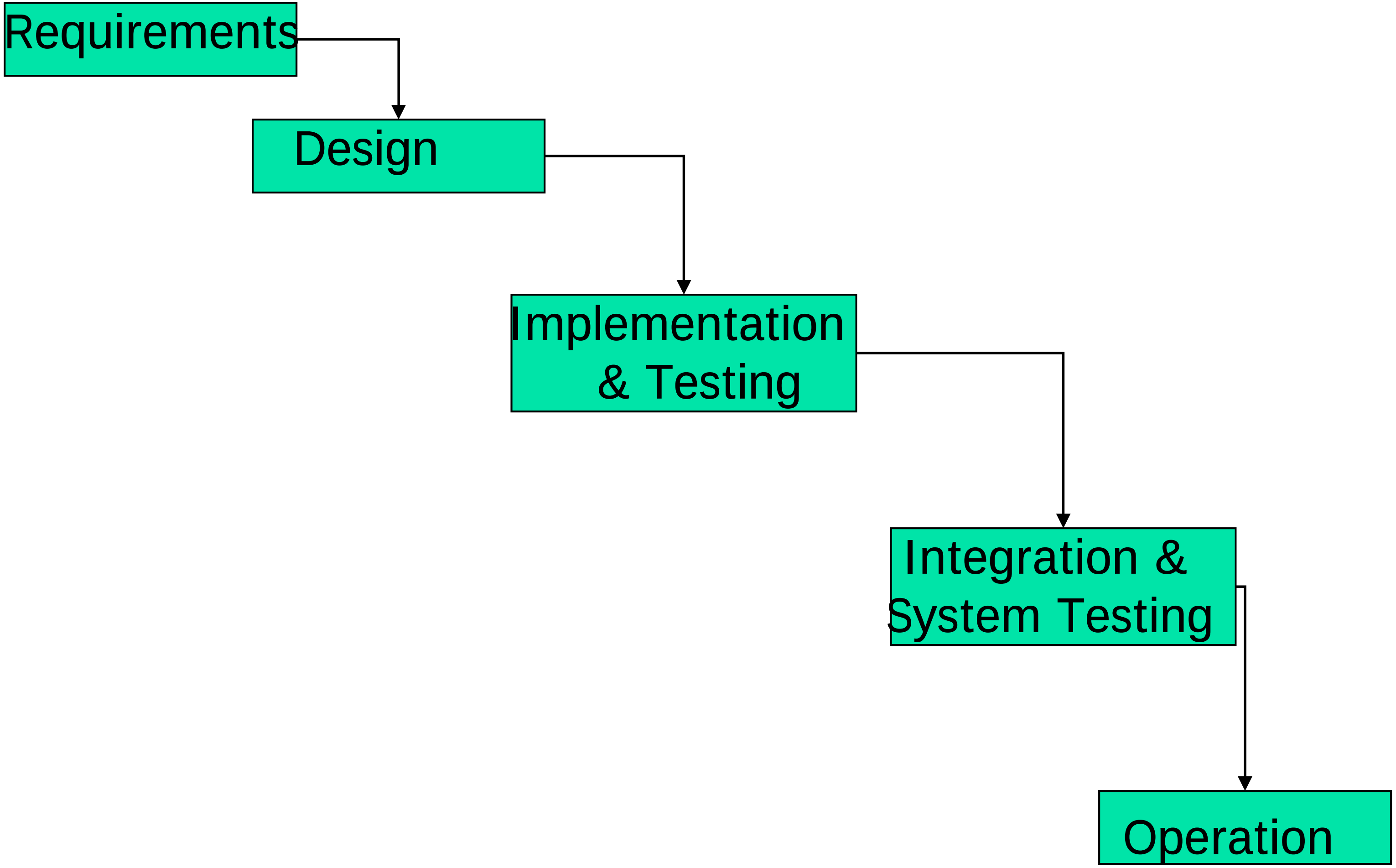
Waterfall Model (1)

- Requirement Analysis phase:
  - One or more meet client (client organization)
- Aim:
  - Determine client needs
  - Current manual system? Understand in-depth
  - Interview potential users
  - What’s client’s way of doing business?
    - (View docs, org charts, procedure manuals)

Waterfall Model

- Each phase must be completed in its entirety before the next phase can begin.
- At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project.
- Phases do not overlap in a waterfall model.

Waterfall Life Cycle



Requirements

Design

Implementation & Testing

Integration & System Testing

Operation

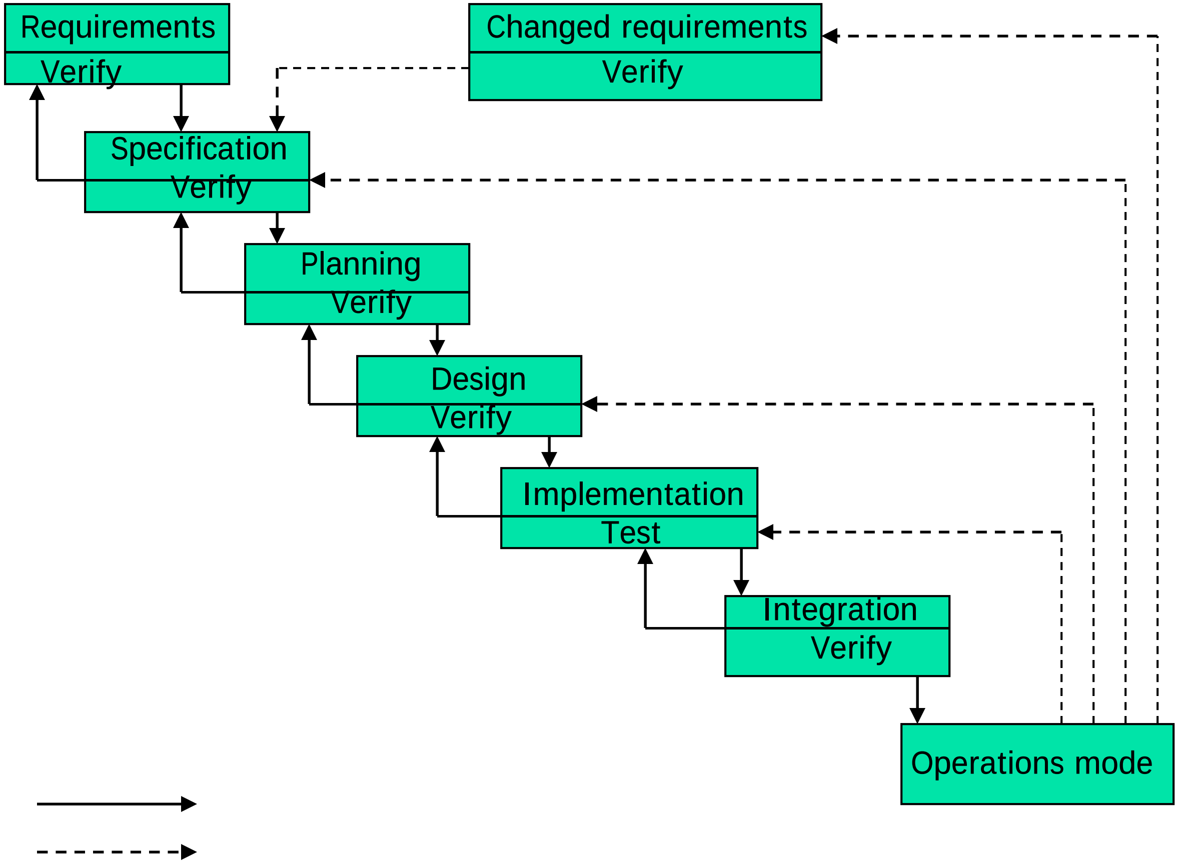
Advantages

- Simple and easy to use.
- Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.

Disadvantages

- Adjusting scope during the life cycle can kill a project
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Poor model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Poor model where requirements are at a moderate to high risk of changing.

Waterfall Model (2)



Waterfall Model (2)

- Verify
  - Check and approve each phase before work on next phase
  - Write requirements document
- Underlying principle:
  - Feedback –

Specification Doc

- State exactly what the product must do
- Must be complete – omit no material fact
- Free of contradictions and ambiguities
- Developer obtains new insights into client's requirements
- Check doc → check by client
- Complete? Now build product satisfying the specs

Requirements Document

1. Introduction
2. Project Goals
3. Major Functions
4. General Outputs
5. General Information Inputs
6. Performance
7. Growth

Requirements Document

8. Operation and Environment
9. Compatibility, Interfaces
10. Reliability, Availability
11. Human Interface
12. Organizational Impact
13. Maintenance and Support
14. Documentation and Training

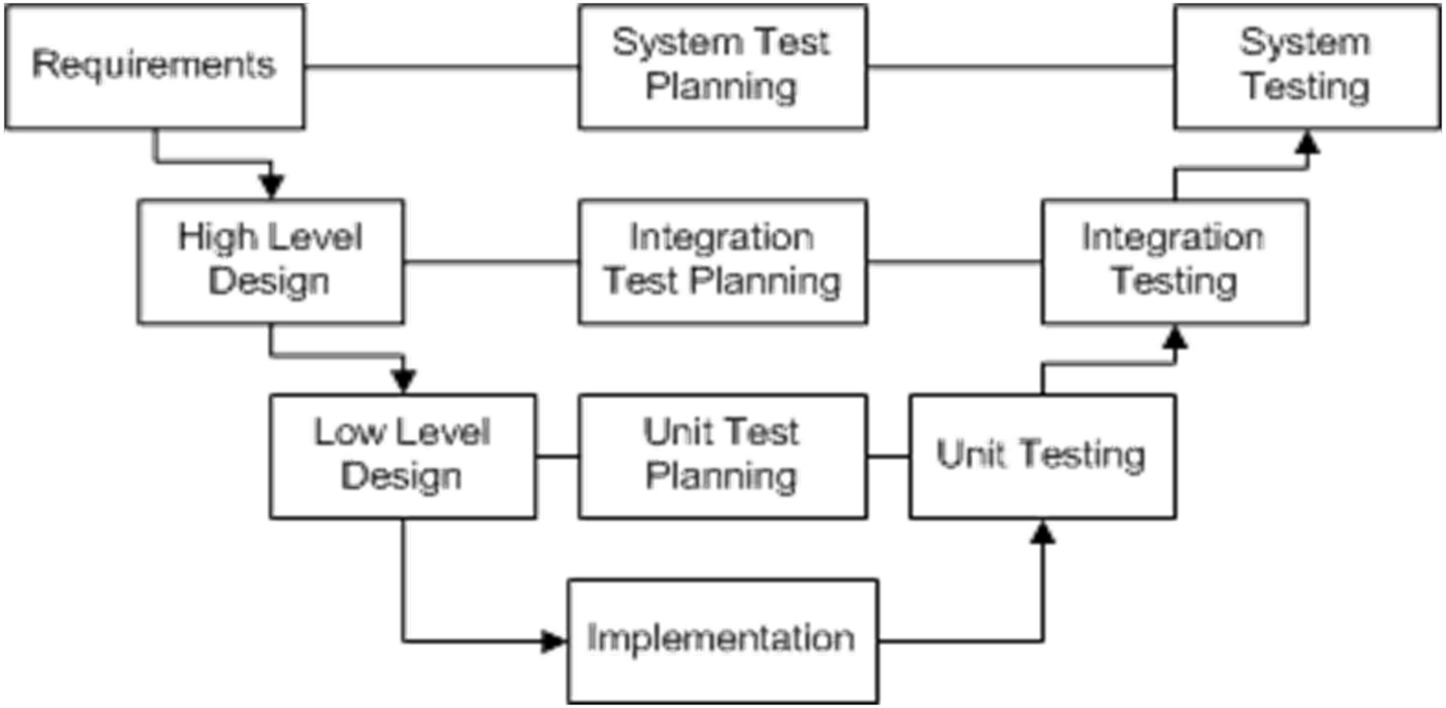
Planning

- Draw up software project management plan (SPMP)
- Major components
  - Describe what must be done
  - How long it will take
  - Cost
  - Human and computer resources needed
  - Detailed time table – who will do what, when
- Client gets projected development time and cost

Planning

- Developer determines:
  - Number of software professionals involved
  - Schedules and deadlines

V-Shaped Model



V-Shaped Model

- A sequential path of execution of processes.
  - Each phase must be completed before the next phase begins.
- Testing is emphasized in this model more so than the waterfall model though.
  - Testing procedures developed early in the life cycle

V-Shaped Model

- Requirements begin the life cycle model
- Before development is started, a system test plan is created.
  - Focuses on meeting the functionality specified in the requirements gathering
- Easy to manage
  - Each phase has specific deliverables
- Phases are processed and completed one at a time.
- Works well for smaller projects

Advantages

- Simple and easy to use.
- Each phase has specific deliverables.
- Higher chance of success
- Works well for small projects

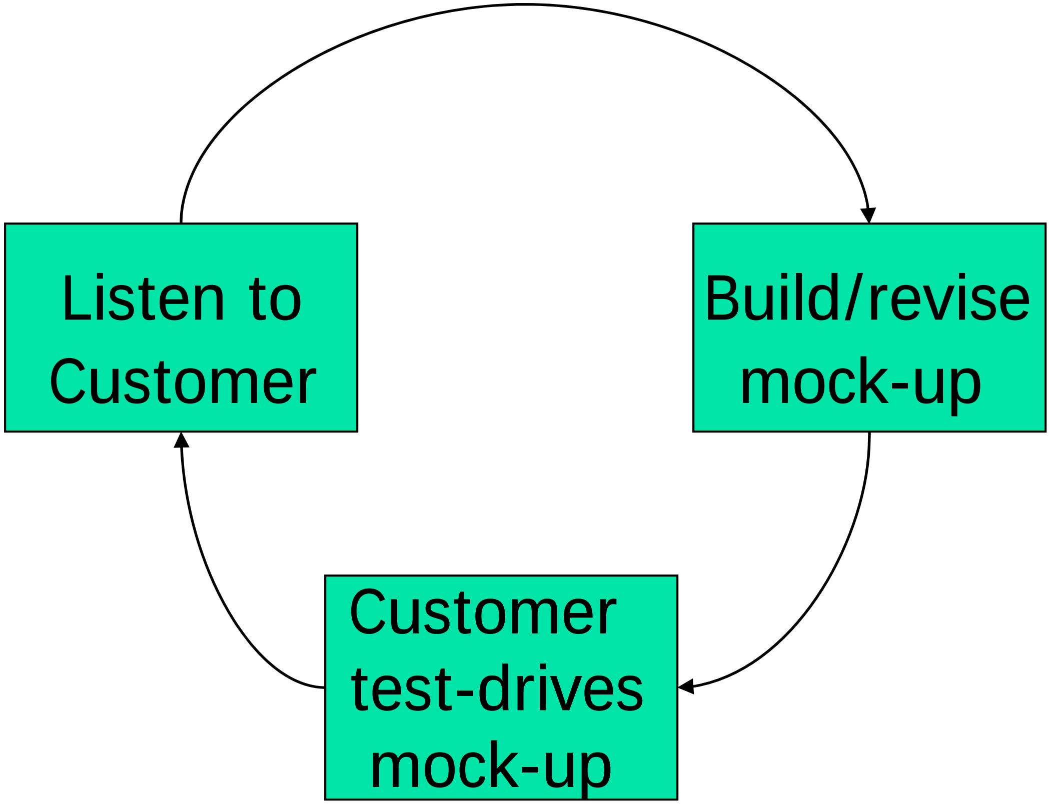
Disadvantages

- Very rigid, like the waterfall model.
- Little flexibility, and adjusting scope is difficult and expensive.
- Software is developed during implementation phase, so no early prototypes produced.
- Model doesn’t provide a clear path for problems found during testing phases.

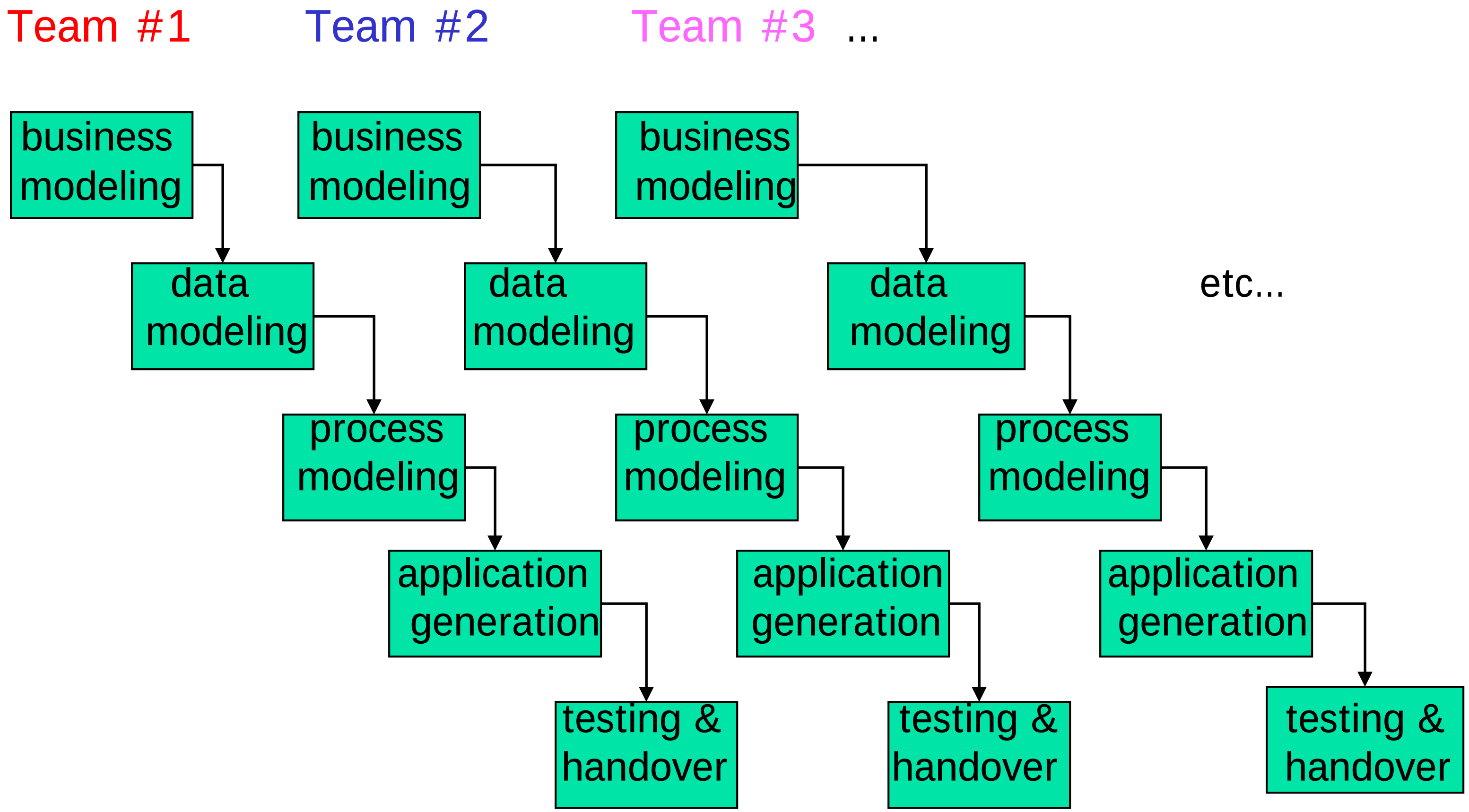
Interactive and Incremental Process

- Modern sw development processes are iterative or incremental
- System modules are refined and transformed through analysis, design, and implementation
  - Details added in successive iterations
  - Changes and improvements introduced as needed
  - Incremental release of software maintain user satisfaction and provide feedback to modules under development

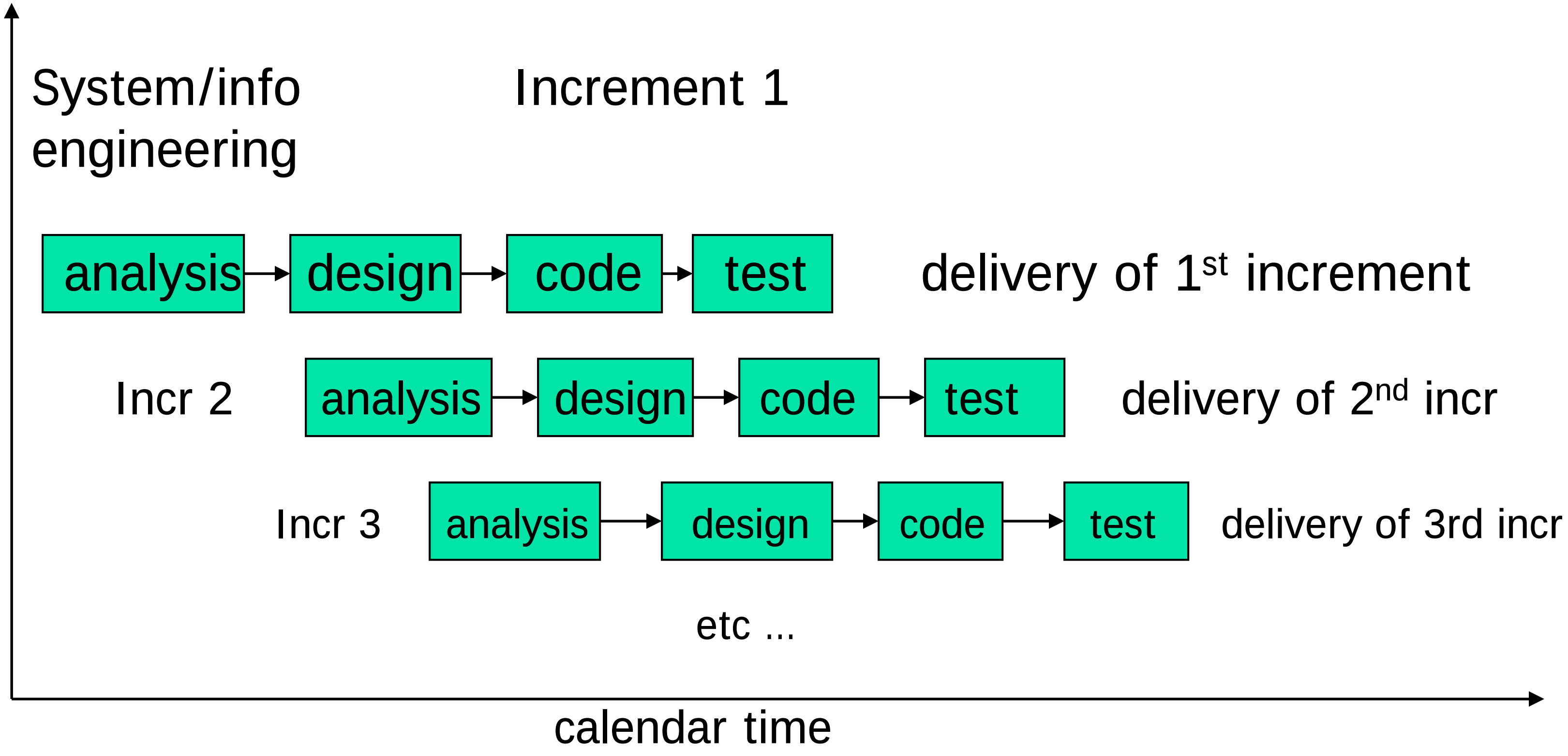
Iterative Models (Prototyping)



Iterative Models (RAD)



The Incremental Model



System/info engineering

Increment 1

analysis → design → code → test delivery of 1st increment

Incr 2 analysis → design → code → test delivery of 2nd incr

Incr 3 analysis → design → code → test delivery of 3rd incr

etc ...

calendar time

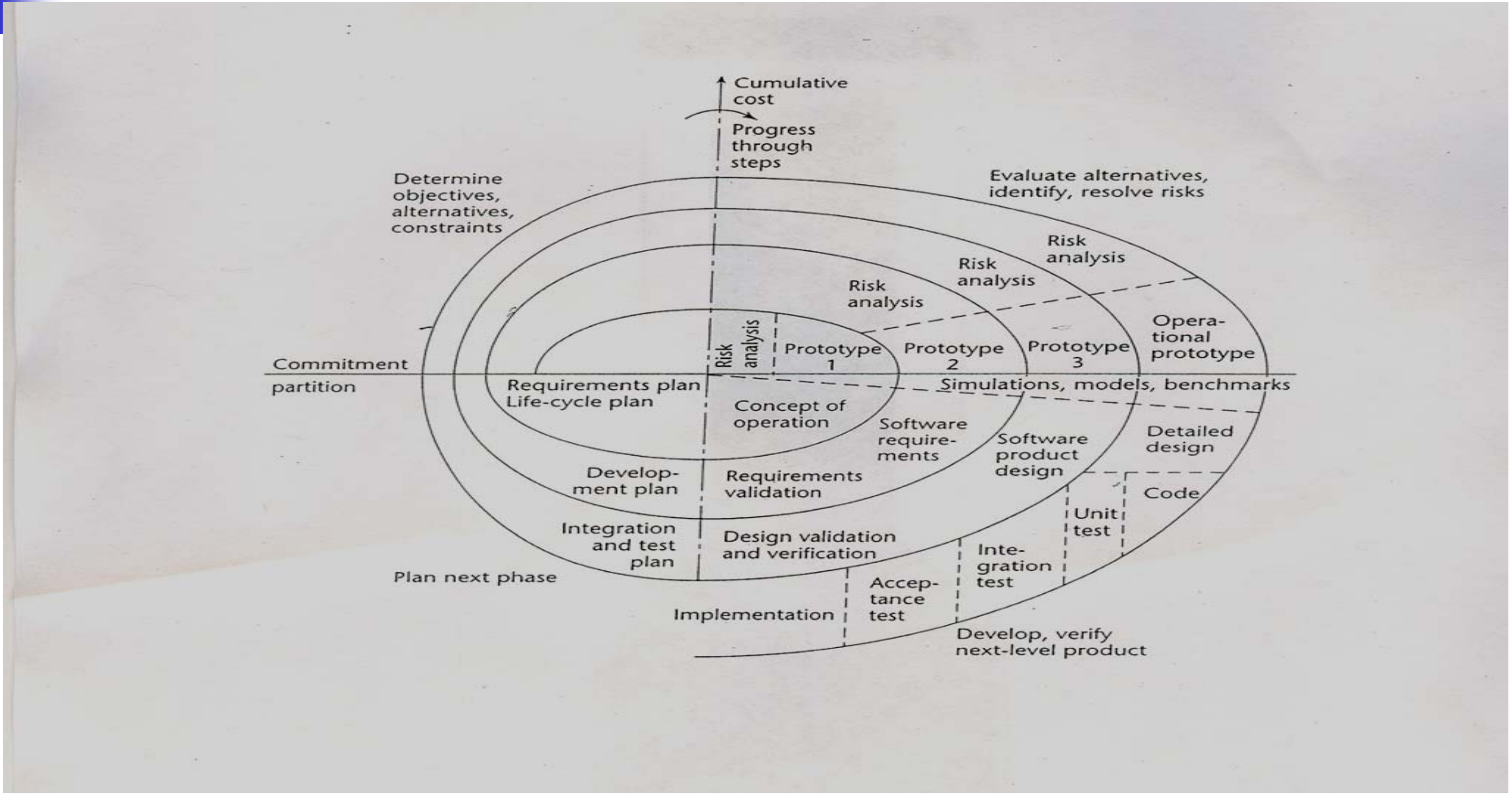
Advantages

- Generates working software quickly and early
- More flexible – less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Easier to manage risk
- Each iteration is an easily managed milestone.

Disadvantages

- Each phase of an iteration is rigid and do not overlap each other.
- Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

Evolutionary (Spiral) Model



Spiral Model

- Risk-based
  - Major objective is to determine the risk involved in product development and resolve each risk in turn
  - Ex 1: delivered product does not satisfy client’s needs.. Then build rapid prototype and let client experiment
  - Ex 2: Cost of building product exceed benefits.
    - To solve, do cost-benefit analysis before each phase and terminate if necessary

Spiral Model

- Complete prototype is approved by client, checked by SQA team, then
  - Team investigate risk if project is continued
- Spiral model characterized by performing risk analysis before each life-cycle phase
- Purpose? Identify and resolve possible risks
- Used only if two criteria are met:
  1. Software system must be large
  2. The product is developed internally

Other Process Models

- Component assembly model:
- Concurrent process model:
- Formal methods
- Clean-room software engineering:

Rational Unified Process (RUP)

- An Interactive process involves managing a stream of executable releases
- An incremental process involves continuous integration of the system’s architecture to produce these releases
  - Each new release embodying incremental improvements over the other

Rational Unified Process

- Success of an interactive and incremental model is predicated on early id of system’s architectural modules
- Modules should
  - Be highly cohesive
  - Have minimal coupling

Rational Unified Process

- Implementation is important
- Unless iterative and incremental development is planned and controlled, it can denigrate to ‘ad hoc’ hacking with no control over the project’s progress

The Process

- Software engineering;
  - a set of procedures, methods, and tools used

- Total Quality Management (TQM):

Stages

- Series of phases termed software life cycle
- Requirements phase (analysis)

- Specification phase
  - Specification document ...