SOFTWARE PROCESS MODELS

Prof. Ralph Laviste
College of Computer Studies
Cebu Institute of Technology
University

OUTLINE

Software Process Models

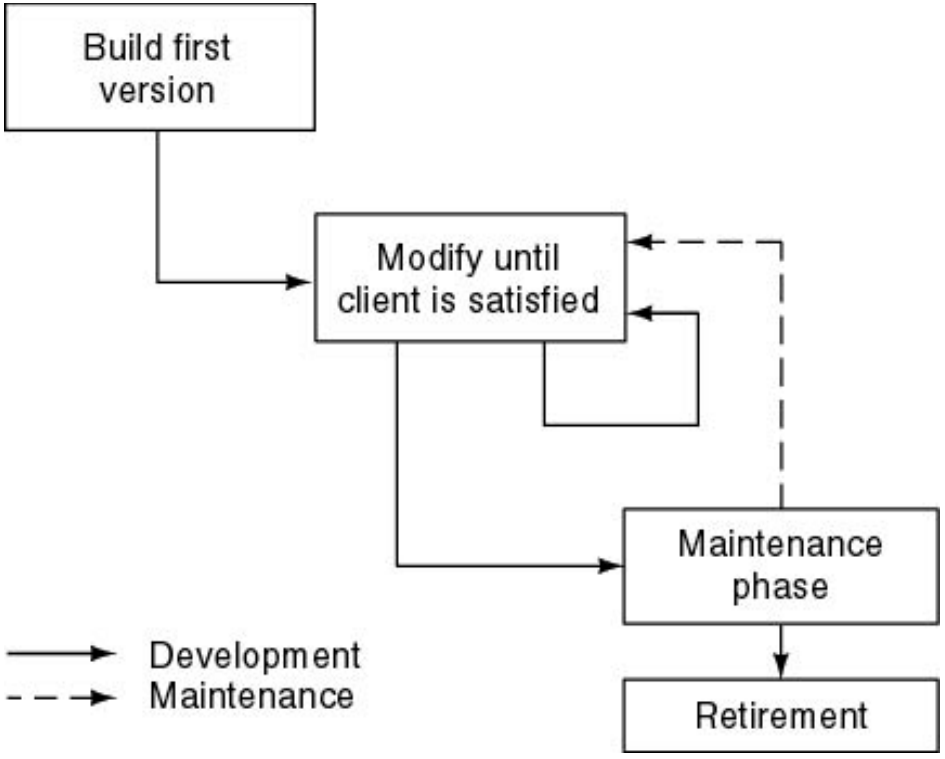
- Build-and-Fix Model
- Waterfall Model
- Incremental Model
- Rapid Prototyping Model
- Spiral Model
- Agile Models
- Object-Oriented Life-Cycle Models

Software Process Model

- An abstract representation of a process.
- Presents a description of a process from some particular perspective.
- The steps through which the product progresses
  - Requirements phase
  - Specification phase
  - Design phase
  - Implementation phase
  - Integration phase
  - Maintenance phase
  - Retirement

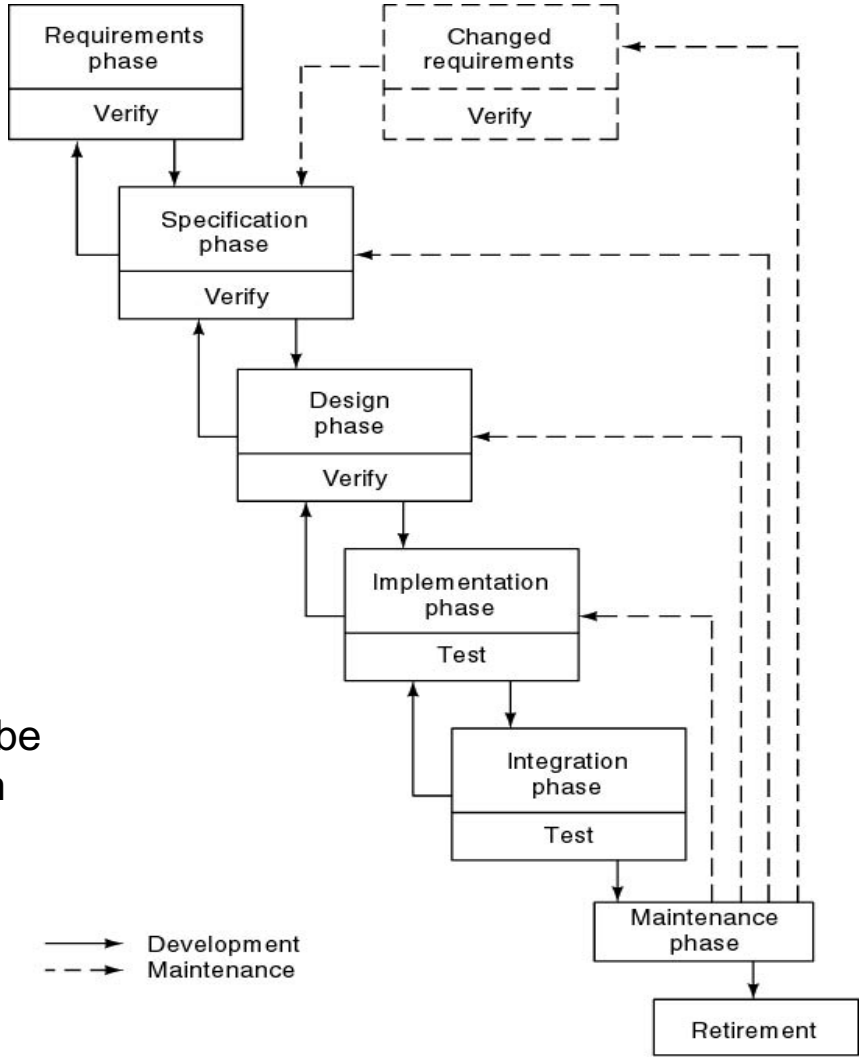
Build and Fix Model

- Characterized by
  - No specifications
  - No design
- Lacks
  - “Game plan”
  - Phases
  - Milestones
- Totally unsatisfactory



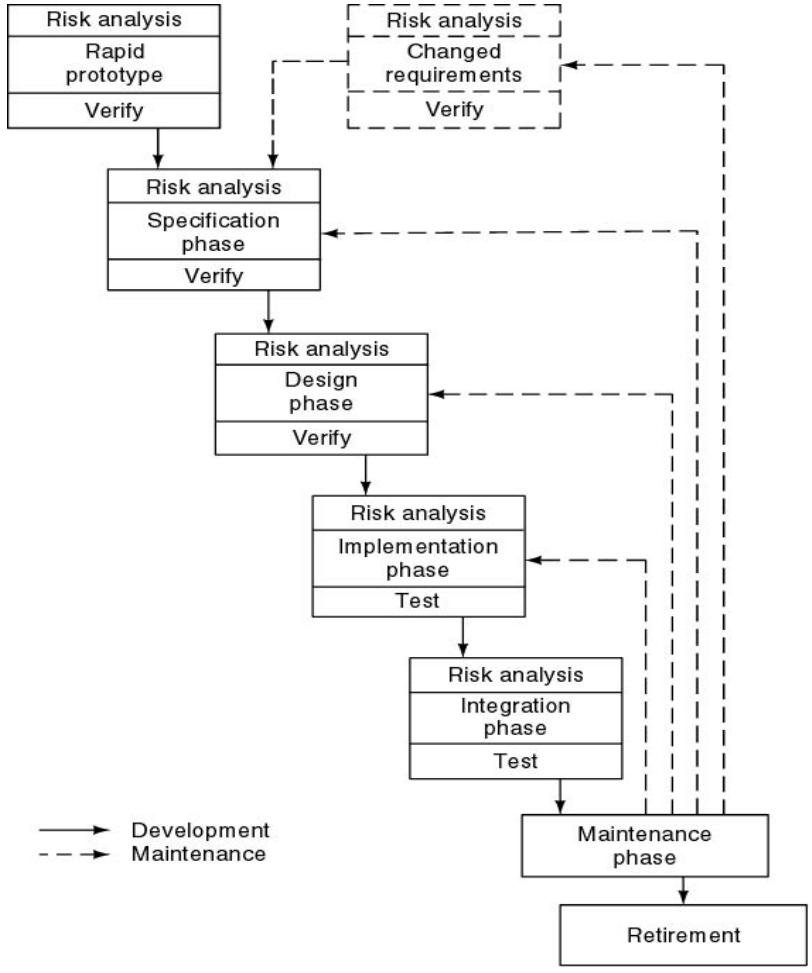
Waterfall Model

- Characterized by
  - Feedback loops
  - Documentation-driven
- Advantages
  - Documentation
  - Easier maintenance
- Disadvantages
  - Try to get it right first time
  - Appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
- Used in large systems engineering projects



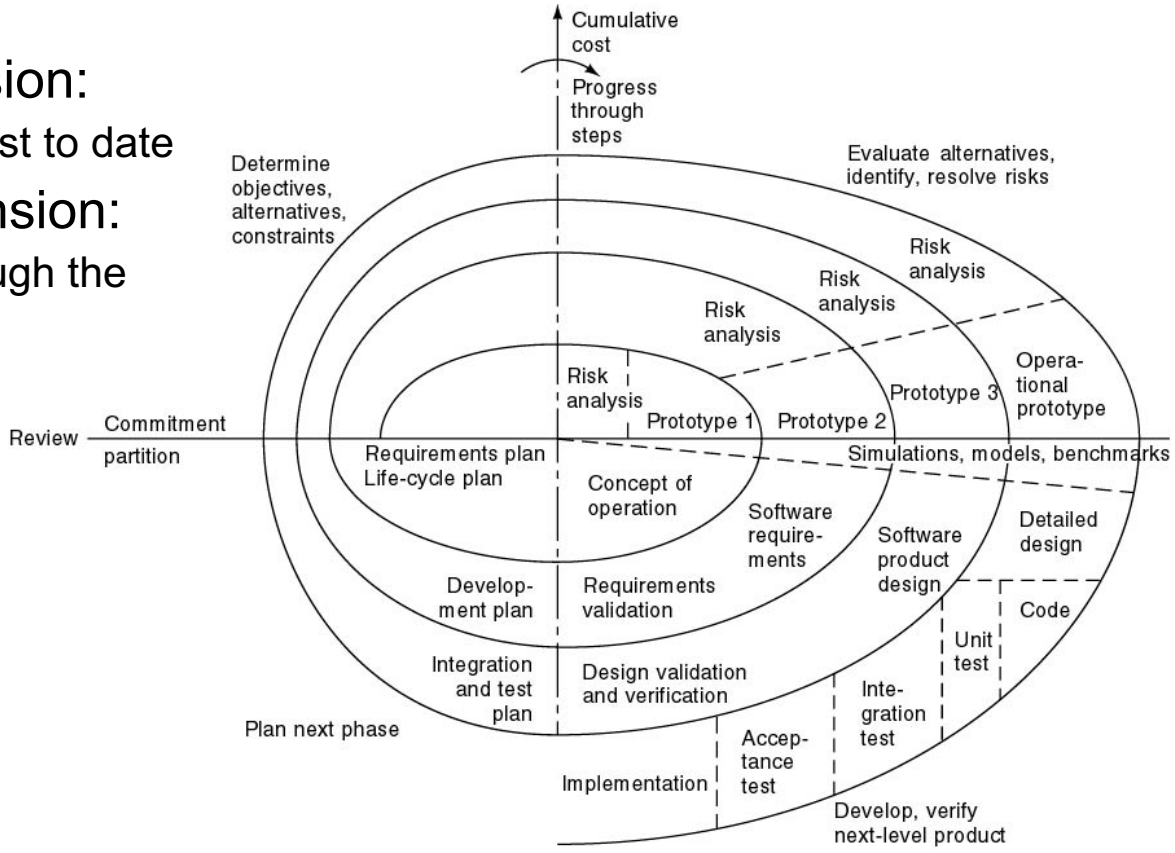
Spiral Model

- Characterized by
  - Waterfall model plus risk analysis. If risks cannot be resolved, project is immediately terminated
  - Precede each phase by alternatives and risk analysis
  - Follow each phase by evaluation and planning of next phase
- Strengths
  - Easy to judge how much to test
- Weaknesses
  - For large-scale software only
  - For internal (in-house) software only



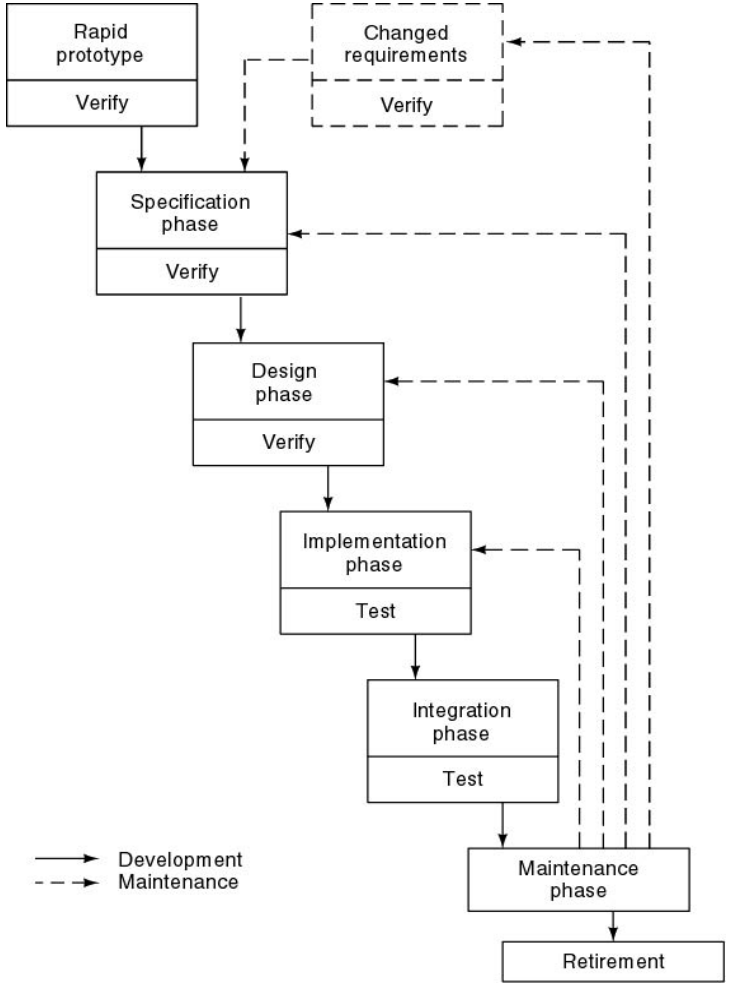
Full Spiral Model

- Radial Dimension:
  - Cumulative cost to date
- Angular Dimension:
  - Progress through the spiral



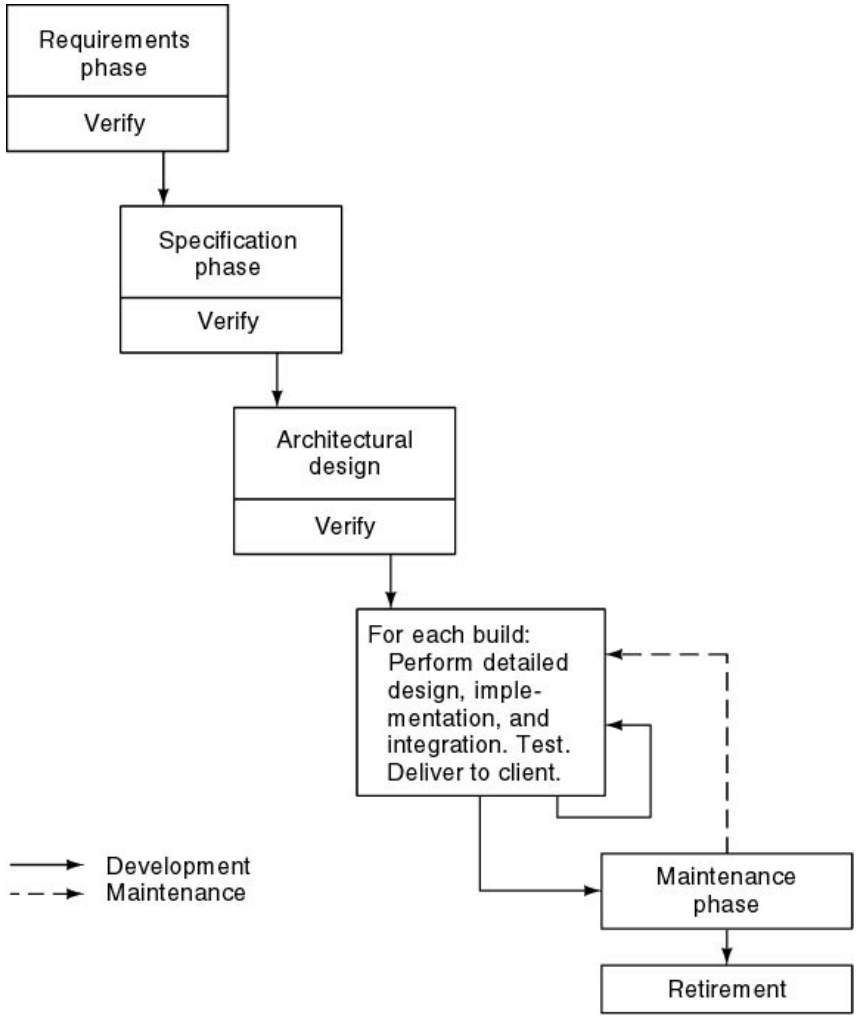
Rapid Prototyping Model

- Characterized by
  - Linear model
  - “Rapid”
  - Frequent change, then discard
- Advantages
  - Works well when client requirements are vague
- Problems
  - Prototype is a “throw-away” system



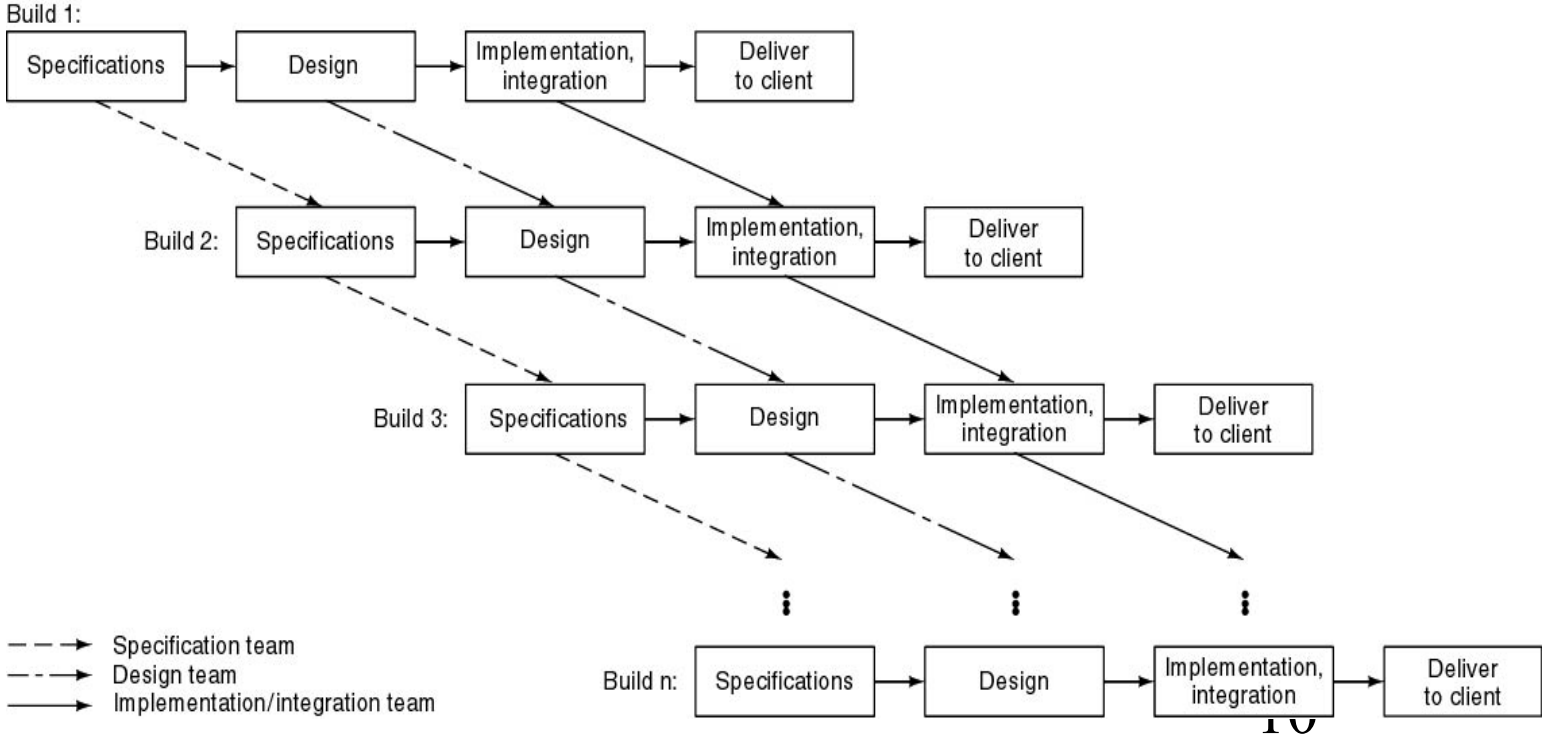
Incremental Model

- Characterized by
  - Project is divided into *builds*
  - Operational quality portion of product within weeks
- Advantages
  - Less traumatic
  - Smaller capital outlay
  - Rapid return on investment
- Problems
  - Need open architecture — maintenance implications
  - Build-and-fix danger



Incremental Model

- More risky version—pieces may not fit
  - CABTAB and its dangers



Synchronize-and Stabilize Model

- Microsoft's life-cycle model
  - Requirements analysis
    - Interview potential customers
  - Draw up specifications
  - Divide project into 3 or 4 builds
  - Each build is carried out by small teams working in parallel
- At the end of the day — synchronize (test and debug)
- At the end of the build — stabilize (freeze build)
- Components always work together
  - Get early insights into operation of product

Agile Models

- Reaction against heavyweight methodologies
  - The crushing weight of corporate *bureaucracy*
  - The rapid pace of information technology *change*
  - The *dehumanizing* of detailed plan-driven development
- Agility (in a software development sense)
  - The ability to respond quickly to change & environment
  - The adaptability to suite new or unexpected challenges
- Where does agility come from?
  - Agile methodologies derive much of their agility by relying on the tacit knowledge embodied in the team, rather than writing the knowledge down in plans.

Agile Alliance

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

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

That is, while there is value in the items on
the right, we value the items on the left more.

Agile Approaches

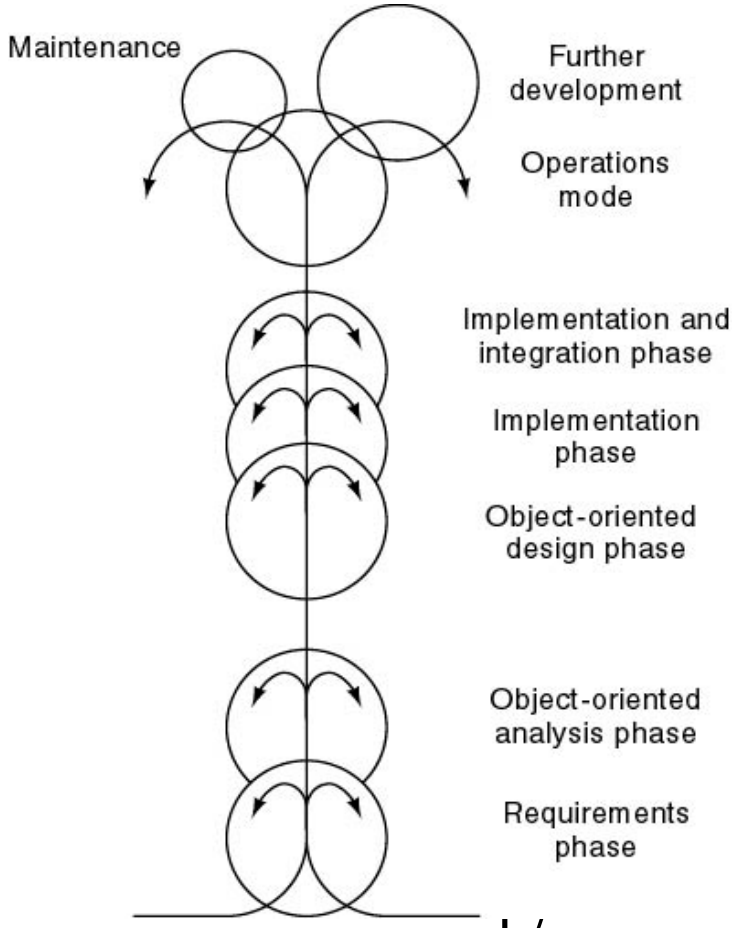
- eXtreme Programming (XP)
- Scrum
- Dynamic Systems Development Method (DSDM)
- Feature-Driven Development (FDD)
- Adaptive Software Development (ASD)
- Crystal Clear
- Agile Modeling
- Unified Software Development Process (USDP) / Rational Unified Process (RUP)
  - Can also be used in an *agile* manner

Object-Oriented Life-Cycle Models

- Characterized by
  - Iteration
  - Parallelism
  - Incremental development
- Danger
  - CABTAB
- Examples
  - Fountain model
  - Recursive/parallel life cycle
  - Round-trip Gestalt
  - Unified Software Development Process

Fountain Model

- Features
  - Overlap (parallelism)
  - Arrows (iteration)
  - Smaller maintenance circle

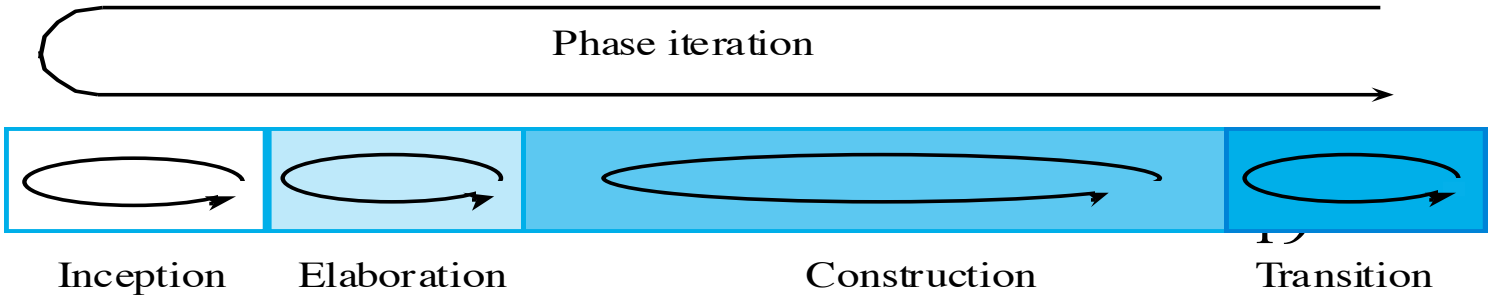


Rational Unified Process (RUP)

- A modern process model derived from the work on the UML and associated process.
- Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A proactive perspective that suggests good practice.
- RUP Good Practice
  - Develop software iteratively
  - Manage requirements
  - Use component-based architectures
  - Visually model software
  - Verify software quality
  - Control changes to software

RUP Phase model

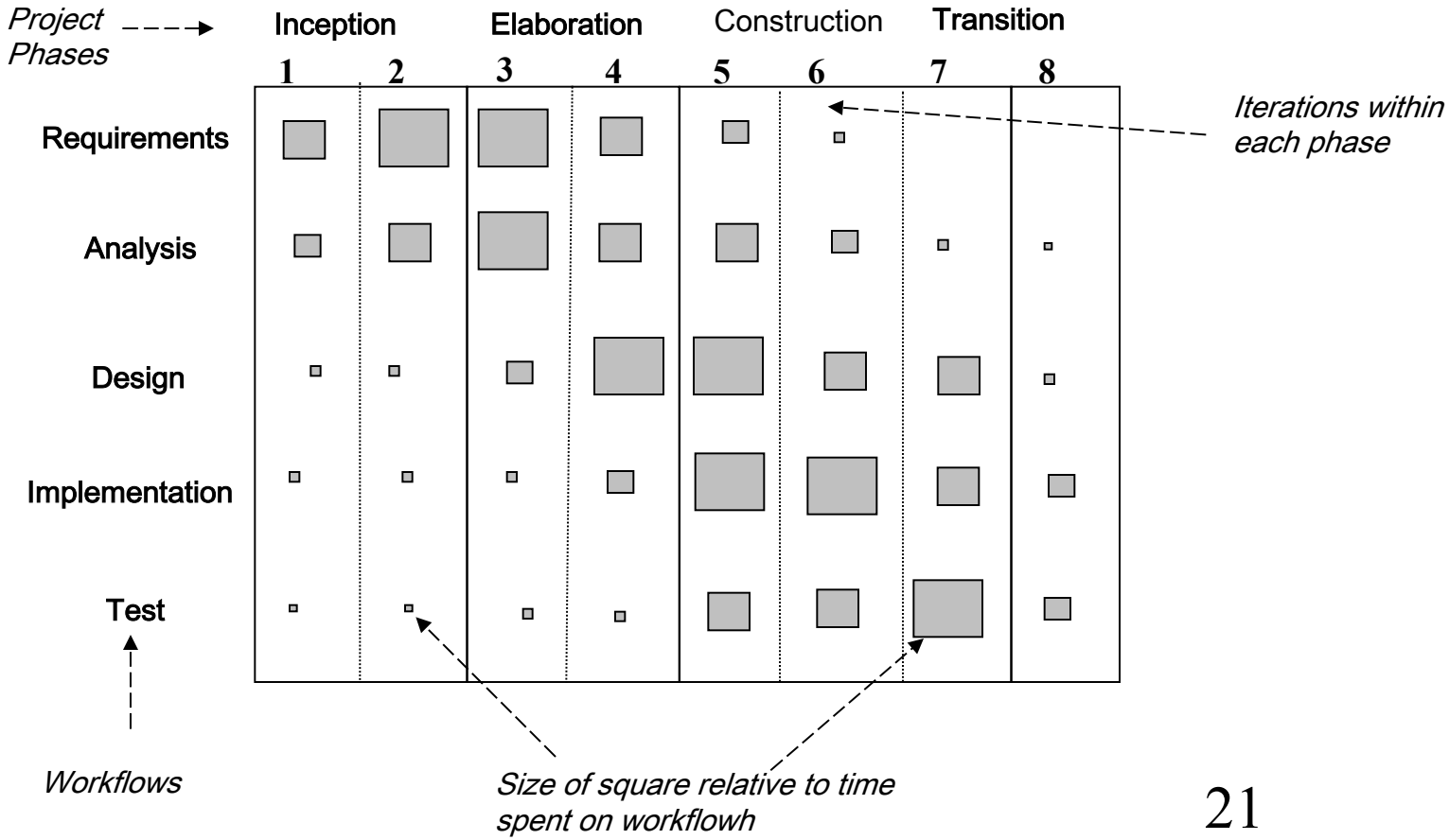
- Inception
  - Establish the business case for the system.
- Elaboration
  - Develop an understanding of the problem domain and the system architecture.
- Construction
  - System design, programming and testing.
- Transition
  - Deploy the system in its operating environment.



USDP

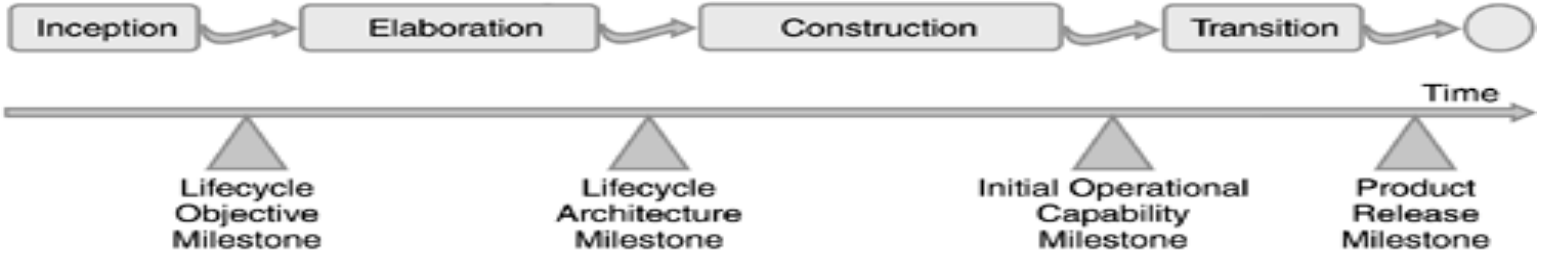
- **U**nified **S**oftware **D**evelopment **P**rocess
  - Public domain methodology for Object-Oriented software development originally developed by the team (in Rational) that created UML
  - Embodies best practices in system development of large and complex systems
  - Led to the famous Rational Unified Process (RUP) which is owned by IBM – the most mature OO methodology to date
- Best Practices
  - Iterative & Incremental development
  - Requirements driven development
  - Architecture centrism
  - Component based development
  - Visual modelling techniques

USDP Model


Project Phases
Inception
Elaboration
Construction
Transition
1
2
3
4
5
6
7
8
Requirements
Analysis
Design
Implementation
Test
Iterations within each phase
Workflows
Size of square relative to time spent on workflowh

USDP

- Inception Phase
  - Understand the scope of the project, Build the business case &
  - Get stakeholder buy-in to move ahead
- Elaboration Phase
  - Mitigate major technical risks, Create a baseline architecture, &
  - Understand what it takes to build the system
- Construction Phase
  - Build the first operational version of the product
- Transition Phase
  - Build the final version of the product and deliver it to the customer



Each milestone is a decision point – begin next phase or stop now?

USDP

Phases ~ Workflows

- Workflows/Activities matter to developers
  - Activities are grouped into workflows, i.e., each workflow consists of a group of activities.
  - Within each phase, activities iterate.
  - Workflows within a phase are the same.
- Phases matter to project managers
  - Phases are sequential, delineated by milestones.
  - Manager's focus shifts from one phase to the next
- The balance of effort spent in each workflow varies from phase to phase.
  - All phases run from requirements to testing, but emphasis changes.
  - At first, main effort is on capture, modeling, analysis of requirements.
  - Later phases emphasise implementation and testing.

USDP

- USDP vs TLC: 2D vs 1D
  - In a TLC project the phases and the workflows/activities are linked together
    - For example, in the *Requirements* phase only *Requirements* workflow activities are carried out; all *Requirements* work should be completed before work starts on *Analysis*.
  - In a USDP project the phases and the workflows/activities are independent with each other
    - For example, some *Requirements* work may be happening alongside *Analysis* work.

Take Home Messages

Software Process Models

- Build-and-Fix Model
- Waterfall Model
- Incremental Model
  - Synchronize-and-Stabilize Model
- Rapid Prototyping Model
- Spiral Model
- Agile
  - eXtreme Programming
- Object-Oriented Life-Cycle Models
  - Fountain Model & USDP / RUP