SOFTWARE PROCESS MODELS

Software Process Models

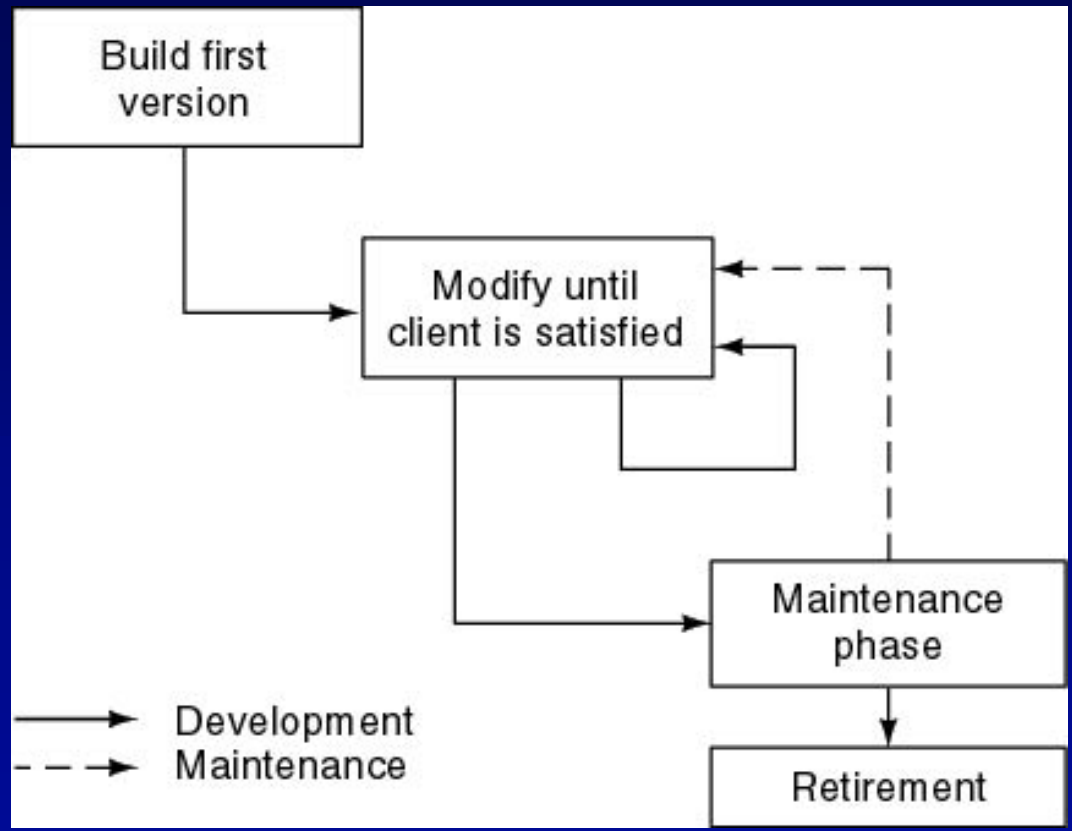
- Process model (Life-cycle model) - steps through which the product progresses
  - Requirements phase
  - Specification phase
  - Design phase
  - Implementation phase
  - Integration phase
  - Maintenance phase
  - Retirement

Overview

- Different process models
  - Build-and-fix model
  - Waterfall model
  - Incremental model
  - Evolutionary process models
    - Rapid prototyping model
    - Spiral model
  - Agile process models
    - Extreme programming
  - Synchronize-and-stabilize model
  - Object-oriented life-cycle models
    - Fountain Model
    - Unified Process

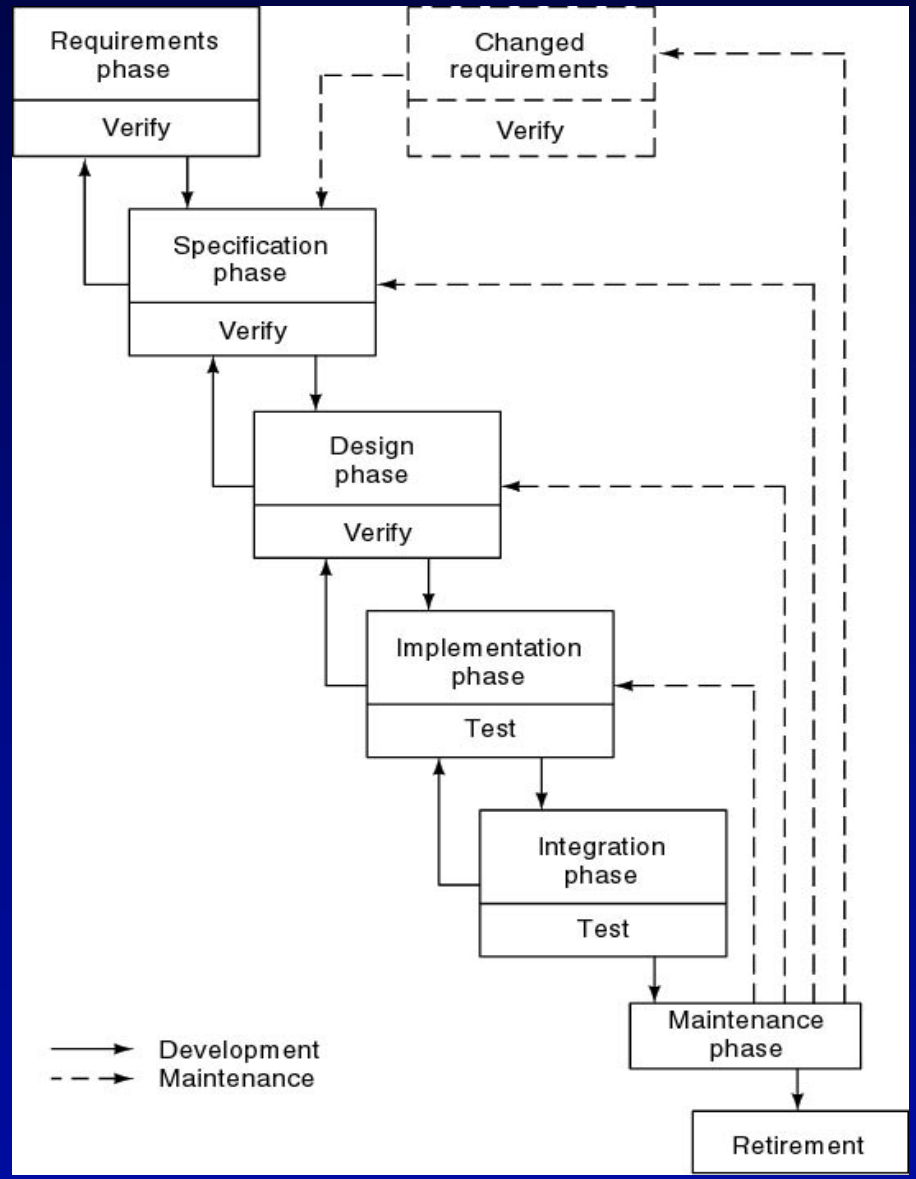
Build-and-Fix Model

- Problems
  - No specifications
  - No design
- Totally unsatisfactory
  - High cost
  - Difficult maintenance



Waterfall Model

- Only model widely used until early 1980s
- Characterized by
  - Feedback loops
  - Documentation-driven

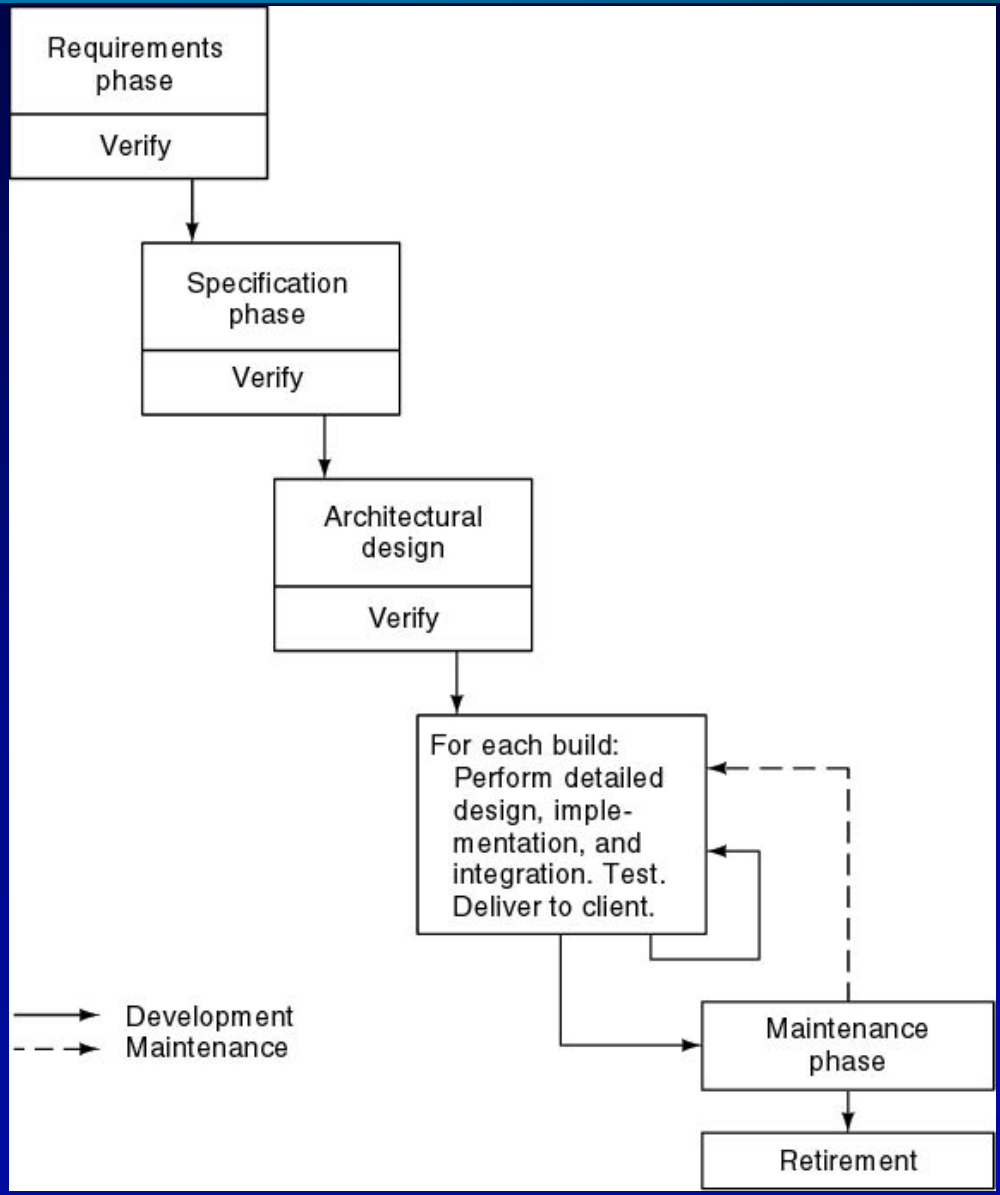


Waterfall Model (contd)

- Advantages
  - Enforces disciplined approach
    - Documentation for each phase
    - Products of each phase checked by SQA group
  - Maintenance is easier
    - Every change reflected in the relevant documentation
- Disadvantages
  - Working version of the software will not be available until late in the project time-span
  - Specifications are long, detailed, written in a style unfamiliar to the client
  - “Blocking states” – some project team members must wait for other team members to complete dependent tasks

Incremental Model

- Divide project into builds – modules interacting to provide a specific functionality
- Typical product -
  5 to 25 builds

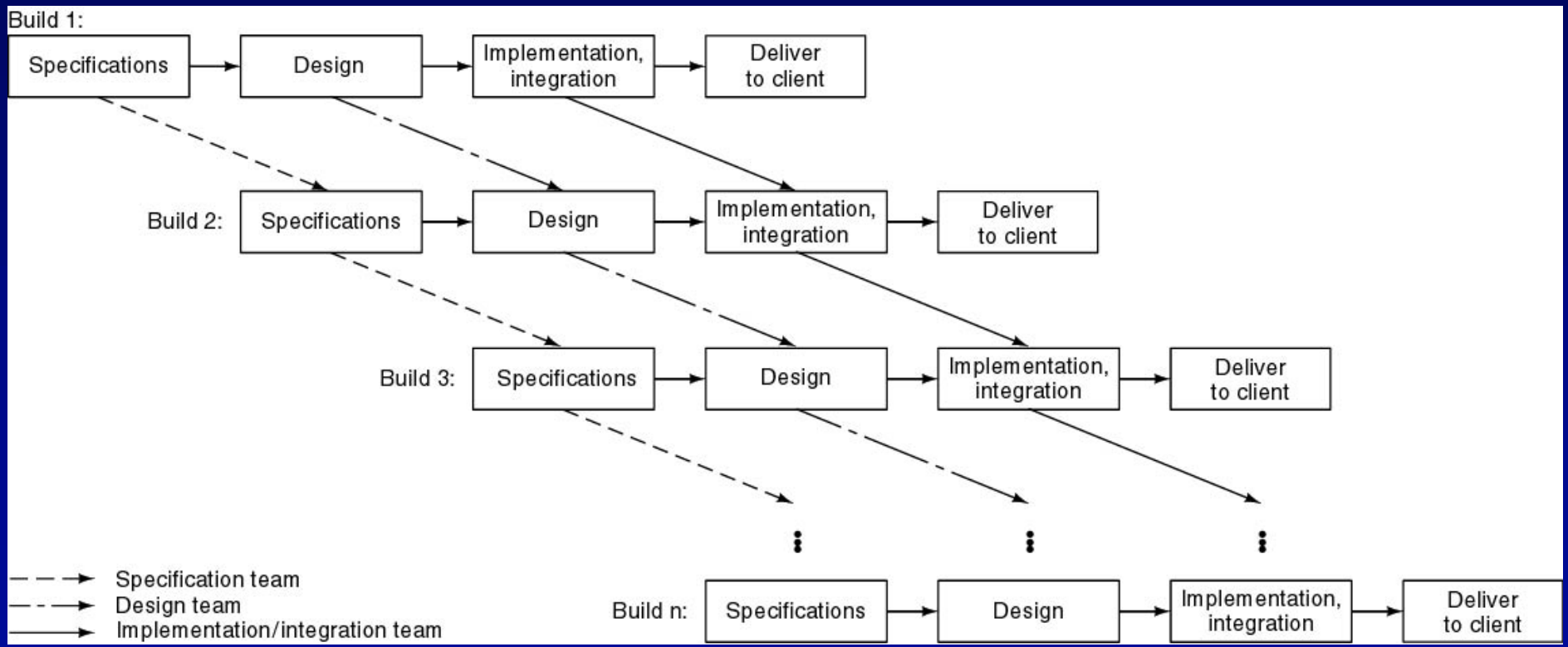


Incremental Model (contd)

- Waterfall and rapid prototyping models
  - Deliver complete product at the end
- Incremental model
  - Deliver portion of the product at each stage
- Advantages
  - Less traumatic
  - Smaller capital outlay, rapid return on investment
  - Open architecture—maintenance implications
- Disadvantages
  - Easily can degenerate into build-and-fix model
  - Contradiction in terms

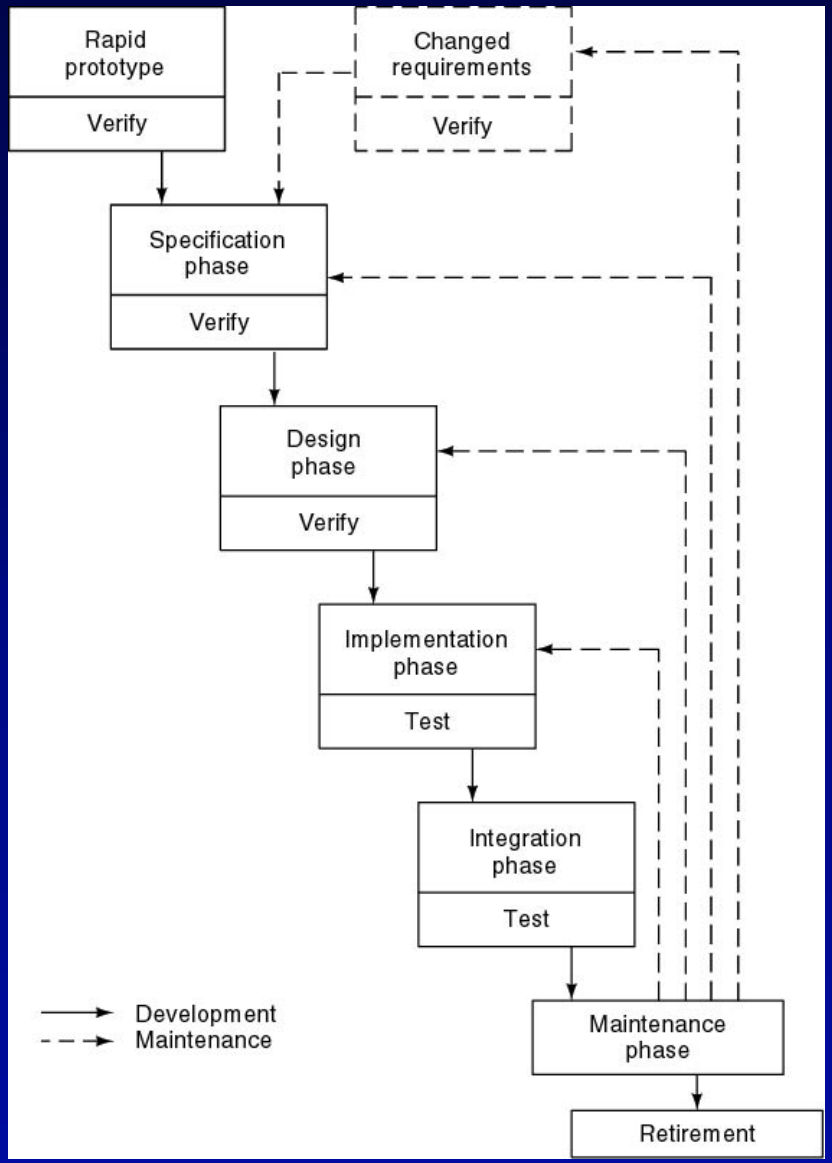
Incremental Model (contd)

- Concurrent incremental model
  - more risky version — pieces may not fit



Rapid Prototyping Model

- First step - construct the prototype as rapidly as possible
  - Only those aspects of the software that will be visible to the customer/user
- Linear model – no feedback loops

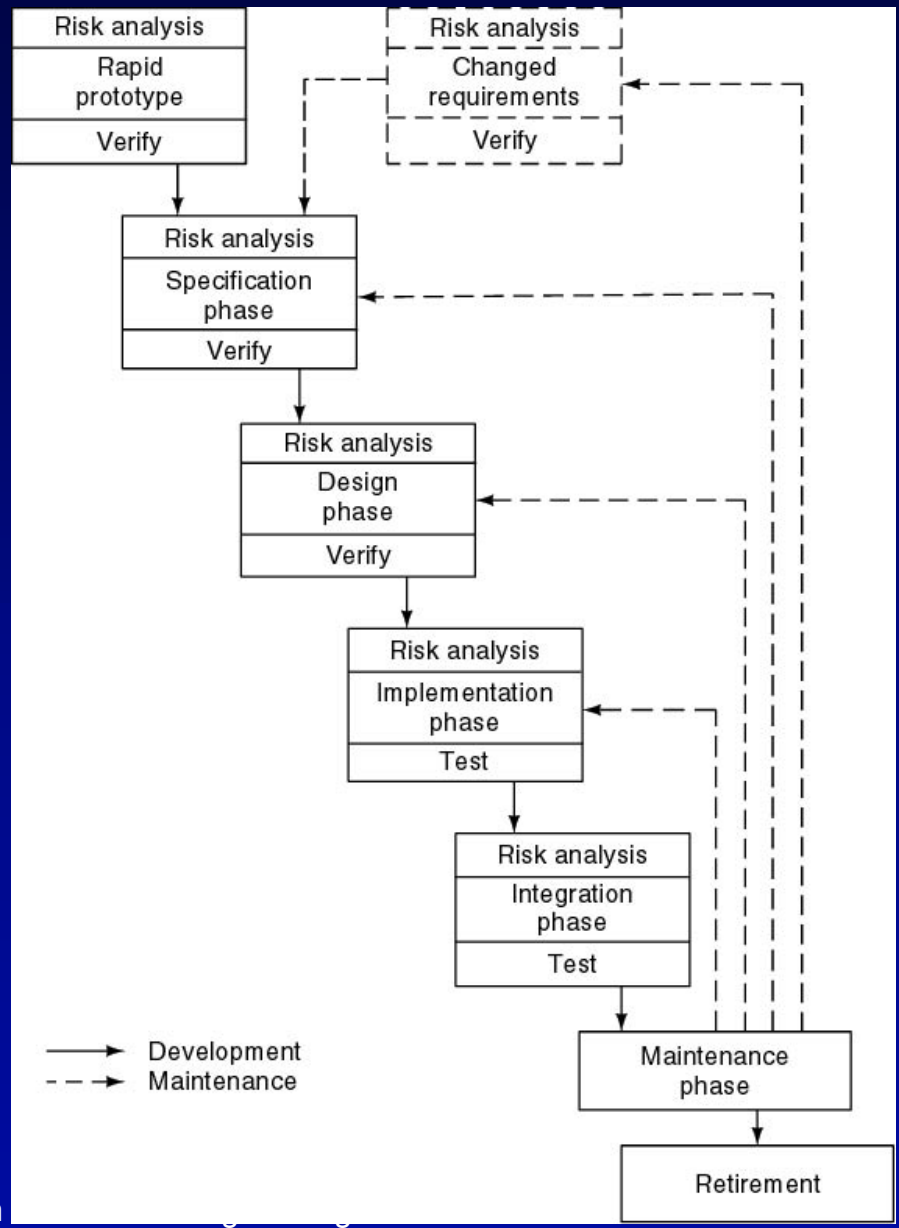


Rapid Prototyping Model (contd)

- Rapid prototype
  - Used in the requirements phase
  - Evaluated by the customer/user
  - Then, it is discarded - do not turn into product
- Rapid prototyping model is not proven and has its own problems
  - Possible solution
    - Rapid prototyping for defining requirements
    - Waterfall model for rest of life cycle

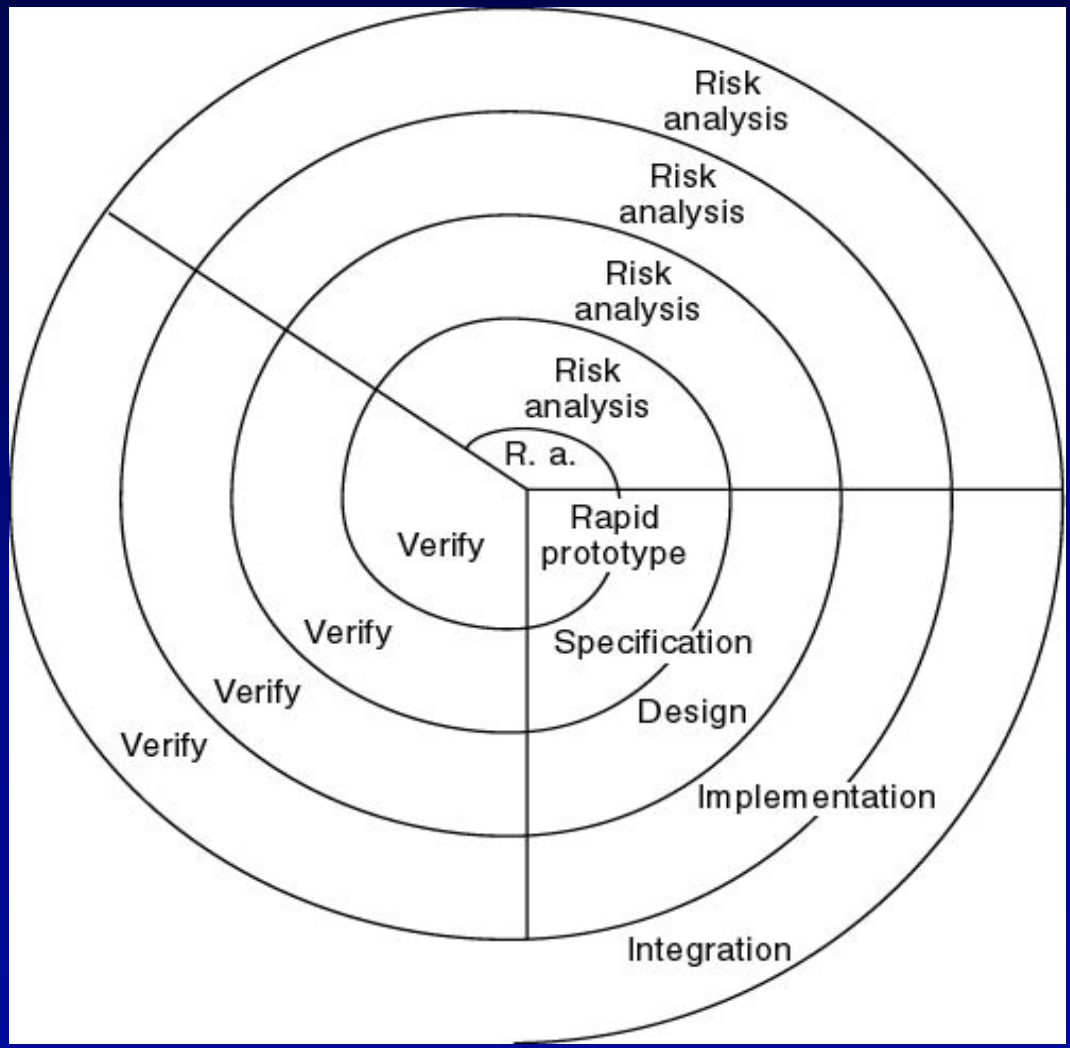
Spiral Model

- Minimize risk via the use of prototype and other means
  - Simplified form - Waterfall model plus risk analysis
- Precede each phase by
  - Alternatives
  - Risk analysis
- Follow each phase by
  - Evaluation
  - Planning of next phase



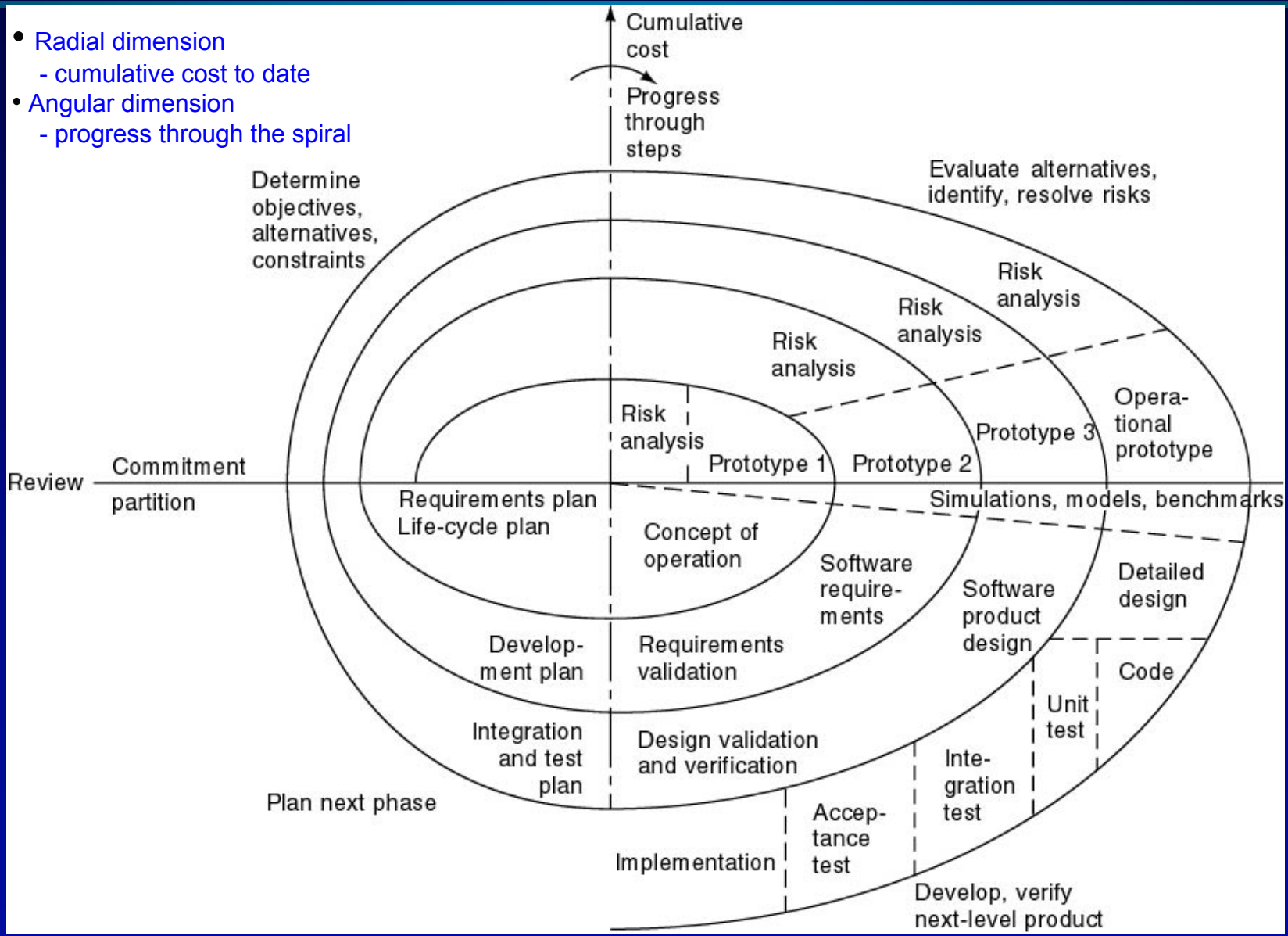
Simplified Spiral Model

- If risks cannot be resolved, project may be terminated immediately



Full Spiral Model (contd)

- Radial dimension
  - cumulative cost to date
- Angular dimension
  - progress through the spiral



Analysis of Spiral Model

- Strengths
  - Answers the question “How much to test ?” in terms of risk
  - No distinction between development and maintenance (another cycle of the model)
- Weaknesses
  - Internal (in-house) development only
    - For contract software, all risk analysis must be performed before the contract is signed, not in the spiral model
  - Large-scale software only
    - For small software performing risk analysis would significantly affect the profit potential

Agile Process Models

- Agile software engineering combines a philosophy and a set of development guidelines
- Philosophy
  - Encourages customer satisfaction and early incremental delivery of the software
  - Small highly motivated project teams
  - Informal methods
  - Minimal software engineering work products
  - Overall development simplicity
- Development guidelines
  - Stress delivery over analysis and design
  - Active and continuous communication between developers and customers

Agile Process Models (contd)

- There are many agile process models
  - Extreme Programming (XP)
  - Adaptive Software Development (ASD)
  - Dynamic System Development Method (DSDM)
  - Scrum
  - Crystal
  - Feature Driven Development (FDD)
  - Agile Modeling (AM)

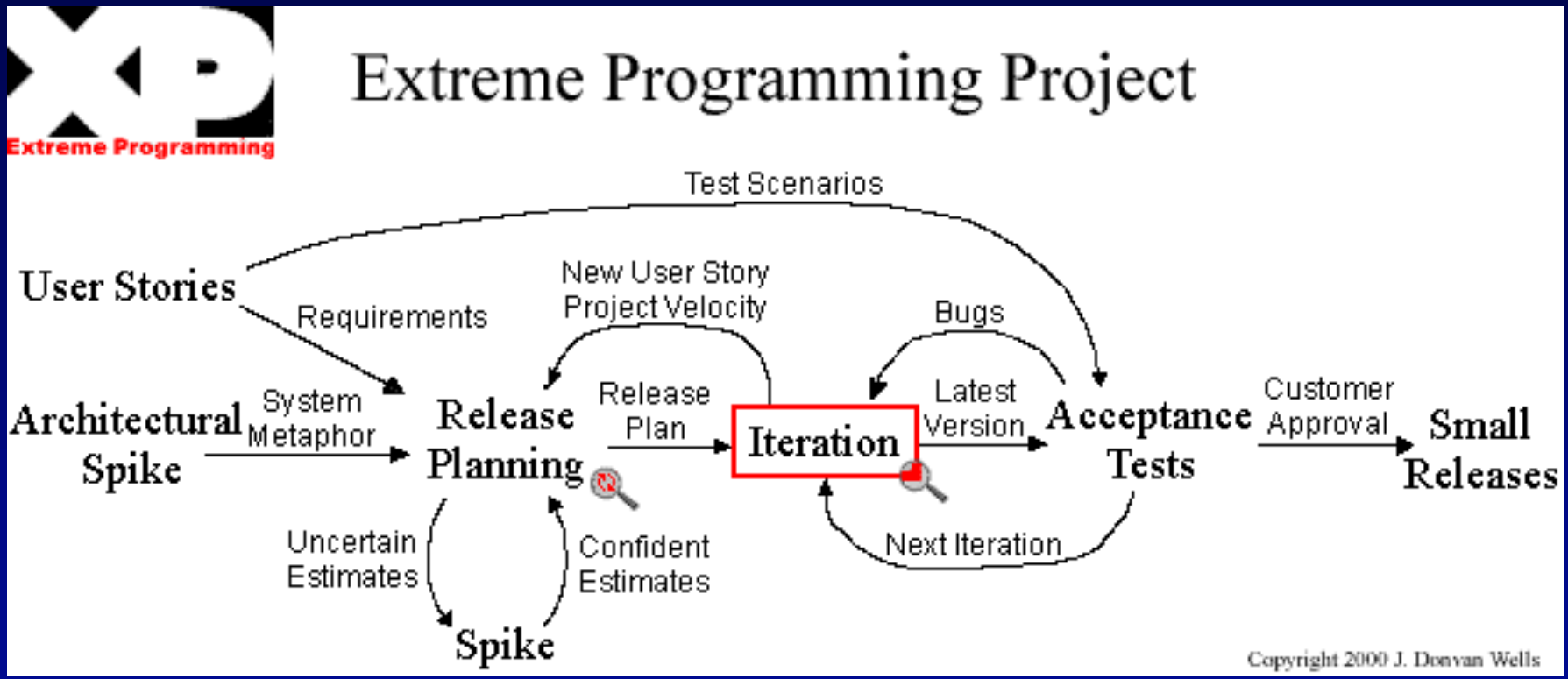
Reading: Choose “Agile Methods” from

http://www.computer.org/portal/site/seportal/index.jsp

Extreme Programming (XP)

- Somewhat controversial new approach; variation of the incremental model
- First step
  - Determine features that client wants (stories)
  - Estimate duration and cost of each feature
- Client selects stories for each successive build
- Each build is divided into tasks
- Test cases for a task are drawn up
- Pair programming – working with a partner on one screen
- Continuous integration of tasks

Extreme Programming (contd)



http://www.extremeprogramming.org/

Features of XP

- Computers are put in center of large room lined with cubicles
- Client representative works with the XP team at all the times
- Individual cannot work overtime for 2 successive weeks
- There is no specialization
  - all members of the XP team work on specification, design, code, and testing
- There is no overall design phase before various builds are constructed – refactoring

Features of XP

- Advantages
  - Useful when requirements are vague or changing
  - Emphasis on teamwork and communication
  - Programmer estimates before committing to a schedule
  - Continuous measurement; frequent, extensive testing
- Disadvantages
  - Limited to small products and small teams - can be disastrous when programs are larger than a few thousand lines of code or when the work involves more than a few people.
  - Lack of design documentation
  - Lack of a structured review process

Synchronize-and-Stabilize Model

- Microsoft's life-cycle model – version of incremental model
- Requirements analysis — interview potential customers; list of features with priorities
- Draw up specifications
- Divide project into 3 or 4 builds
- Each build is carried out by small teams working in parallel

Synchronize-and-Stabilize Model (contd)

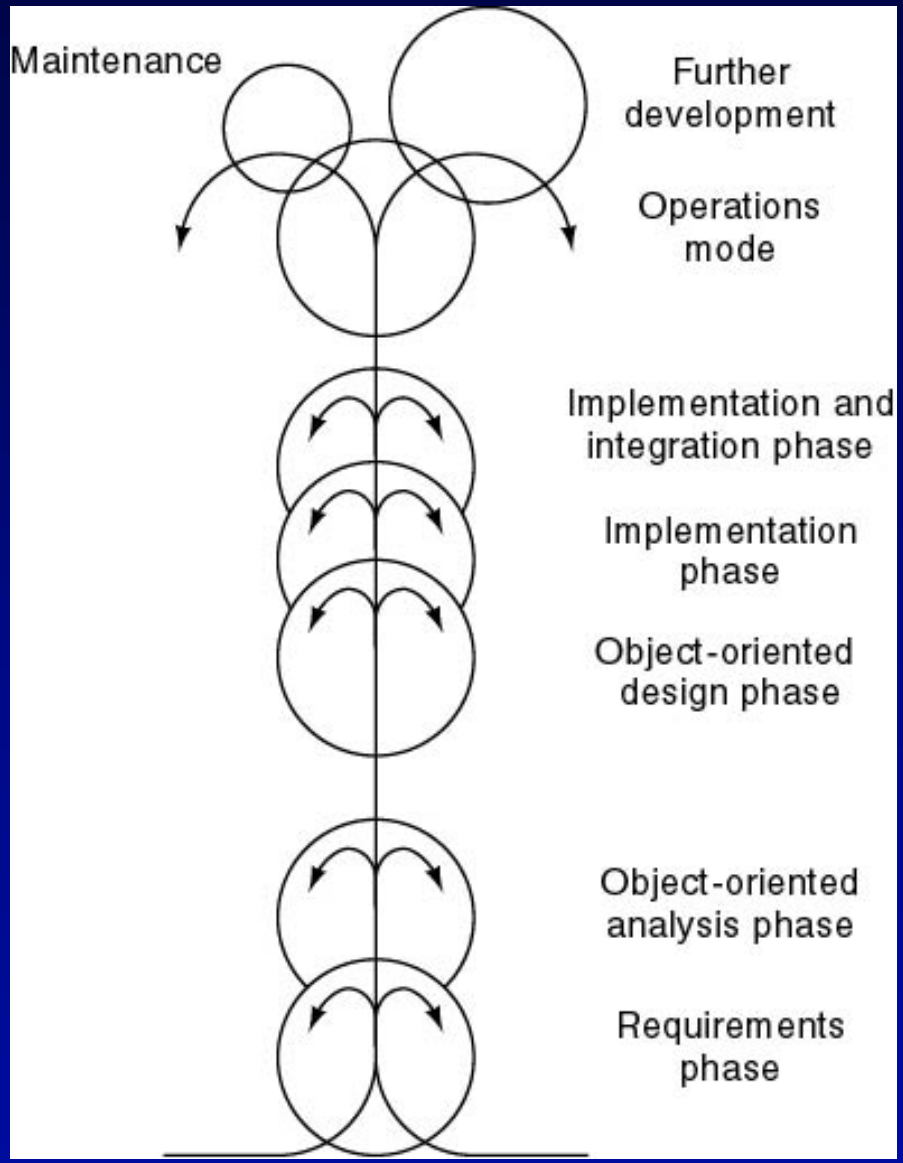
- At the end of the day — synchronize (put together partially completed components; test and debug)
- At the end of the build — stabilize (fix the remaining faults; freeze build)
- Advantages
  - Components always work together
  - Early insights into operation of product
  - Model can be used even the initial specification is incomplete

Object-Oriented Life-Cycle Models

- Need for iteration within and between phases
  - Fountain model
  - Unified software development process
- All incorporate some form of
  - Iteration
  - Parallelism
  - Incremental development

Fountain Model

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



Unified Process

- Unified process is a framework for OO software engineering using UML (Unified Modeling Language)
  - Book by Ivar Jacobson, Grady Booch, and James Rumbaugh (1999)

- Unified process (UP) is an attempt to draw on the best features and characteristics of conventional software process models, but characterize them in a way that implements many of the best principles of agile software development

Unified Process: Phases

- Inception phase
  - Encompasses the customer communication and planning activities
  - Rough architecture, plan, preliminary use-cases
- Elaboration phase
  - Encompasses the customer communication and modeling activities
  - Refines and expands preliminary use-cases
  - Expands architectural representation to include: use-case model, analysis model, design model, implementation model, and deployment model
  - The plan is carefully reviewed and modified if needed
- Construction phase
  - Analysis and design models are completed to reflect the final version of the software increment
  - Using the architectural model as an input develop or acquire the software components, unit tests are designed and executed, integration activities are conducted
  - Use-cases are used to derive acceptance tests

Unified Process: Phases

- Transition phase
  - Software is given to end-users for beta testing
  - User report both defects and necessary changes
  - Support information is created (e.g., user manuals, installation procedures)
  - Software increment becomes usable software release
- Production phase
  - Software use is monitored
  - Defect reports and requests for changes are submitted and evaluated

Unified Process: Major work products

Inception phase

Vision document
Initial use-case model
Initial project glossary
Initial business case
Initial risk assessment
Project plan
  phases and iterations
Business model if necessary
One or more prototypes

Elaboration phase

Use-case model
Supplementary requirements,
  including non-functional
Analysis model
Software architecture description
Executable architectural prototype
Preliminary design model
Revised risk list
Project plan including
  iteration plan
  adapted workflows
  milestones
  technical work products
Preliminary user manual

Unified Process: Major work products

Construction phase

Design model
Software components
Integrated software
    increment
Test plan and procedure
Test cases
Support documentation
    user manuals
    installation manuals
    description of current increment

Transition phase

Delivered software
    increment
Beta test reports
General user feedback

Conclusions

- Different process models, each with its own strengths and weaknesses
- Criteria for deciding on a model include
  - Organization
  - Its management style
  - Skills of the employees
  - Product nature