Software Engineering

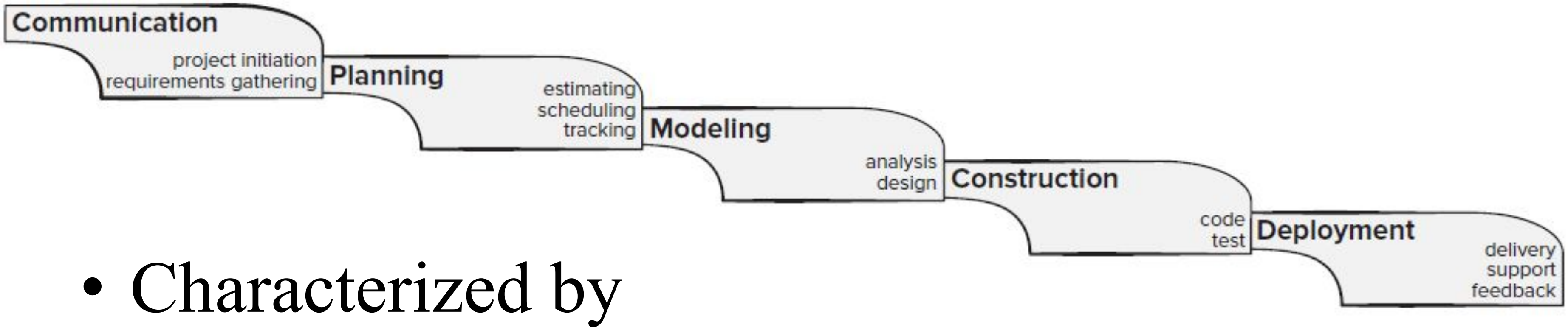
Process Models

Dr. Farrukh Zeshan

Software Life-Cycle Models

- The way you organize your activities
- The steps through which the product progresses
  - Requirements phase
  - Specification phase
  - Design phase
  - Implementation phase
  - Integration phase
  - Maintenance phase
  - Retirement

Waterfall Model



- Characterized by
  - Feedback loops
  - Documentation-driven
- Advantages
  - Documentation
  - Maintenance easier

Pro's and Cons of the Waterfall Model

- **Pro's:**
  - Imposes structure on complex projects
  - Every stage needs to be checked and signed off:
    - Elimination of midstream changes
  - Good when quality requirements dominate cost and schedule requirements

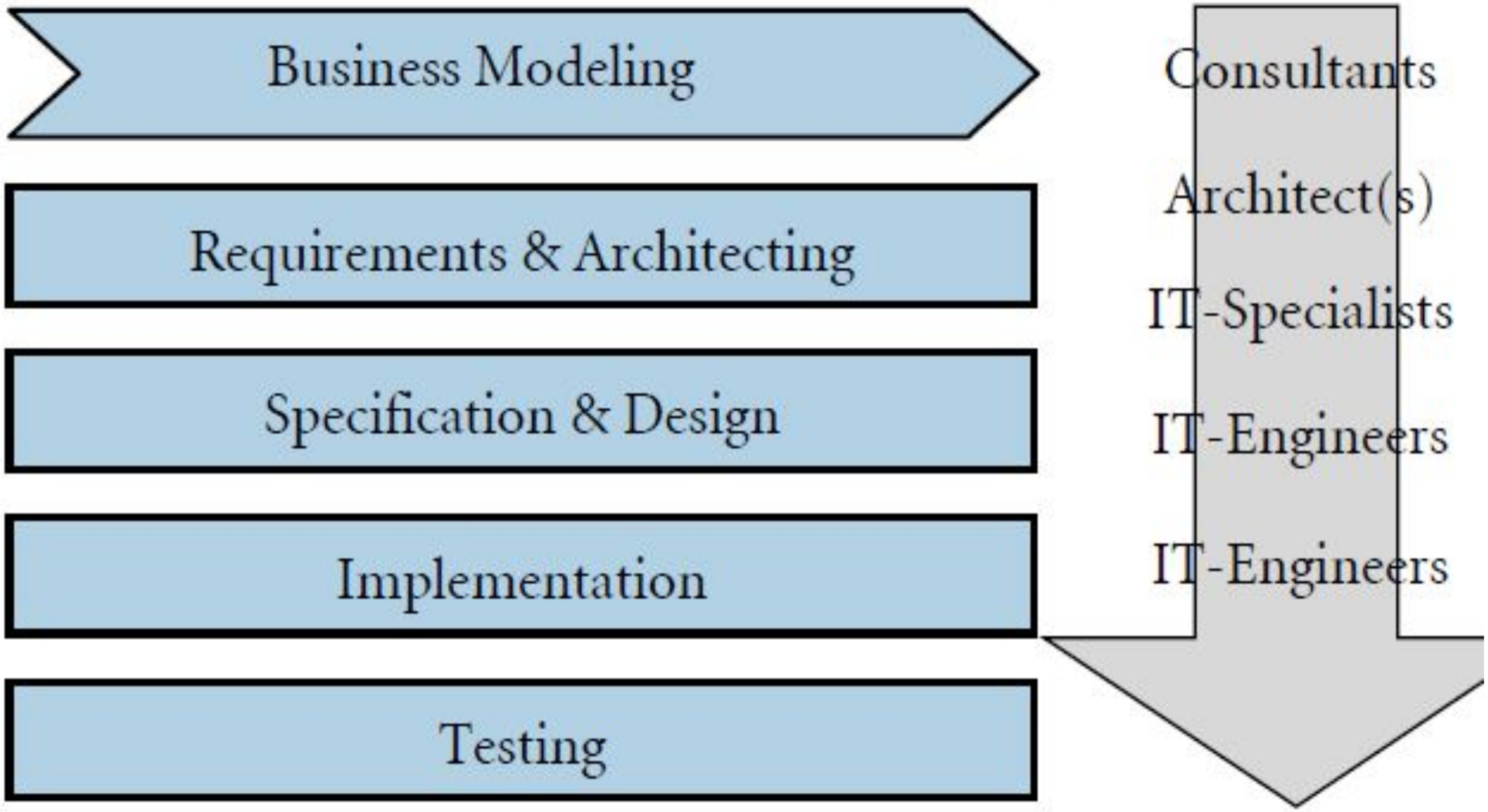
- **Cons:**
  - Limited scope for flexibility / iterations
  - Full requirements specification at the beginning:
  - User specifications
  - No tangible product until the end

Waterfall Model

- Idealized, doesn’t match reality well.
- Unrealistic to expect accurate requirements so early in project.
- Software is delivered late in project, delays discovery of serious errors.
- Difficult to integrate risk management.
- Difficult and expensive to make changes to documents, ”swimming upstream”.

Problems of the Waterfall Process (2)

Different phases are handled by different people



Communication becomes highly critical

Waterfall Model

The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.

- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
- Few business systems have stable requirements.

The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

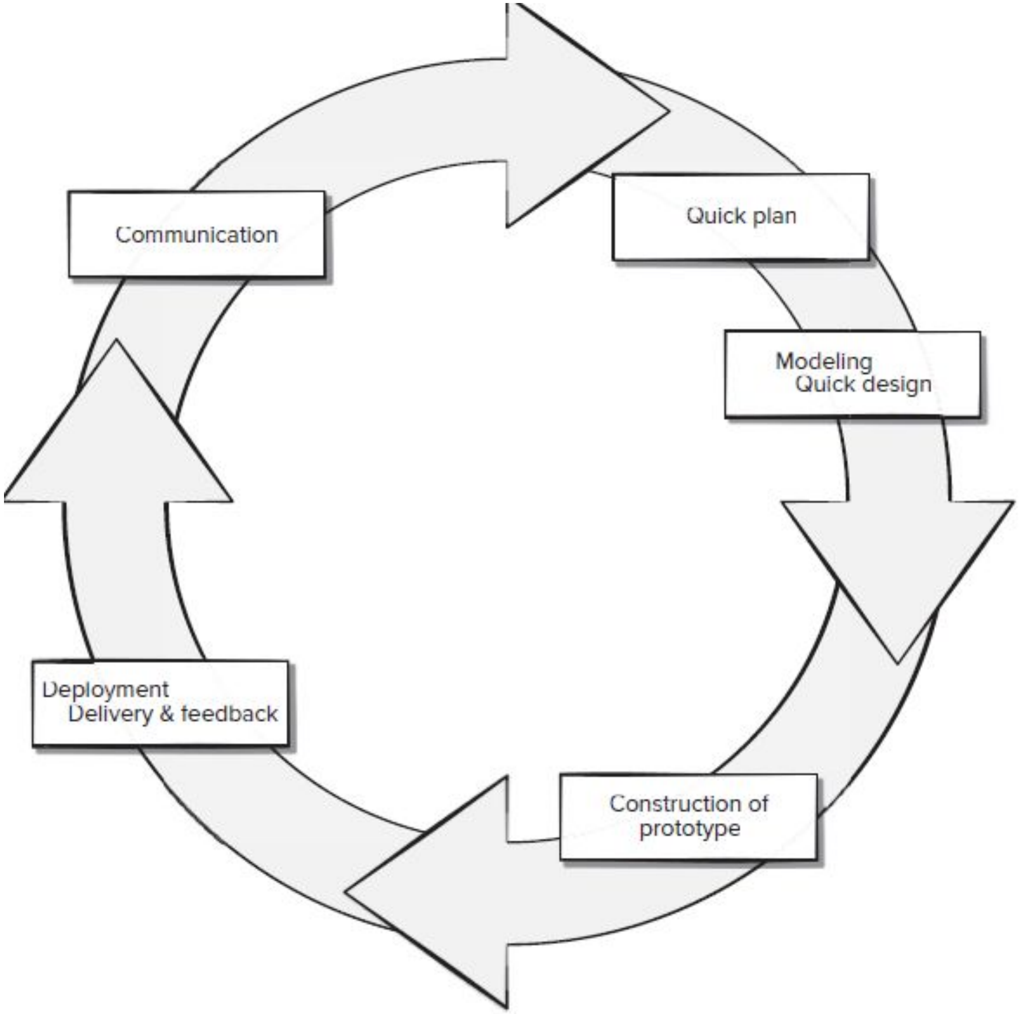
- In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Use Waterfall Model When

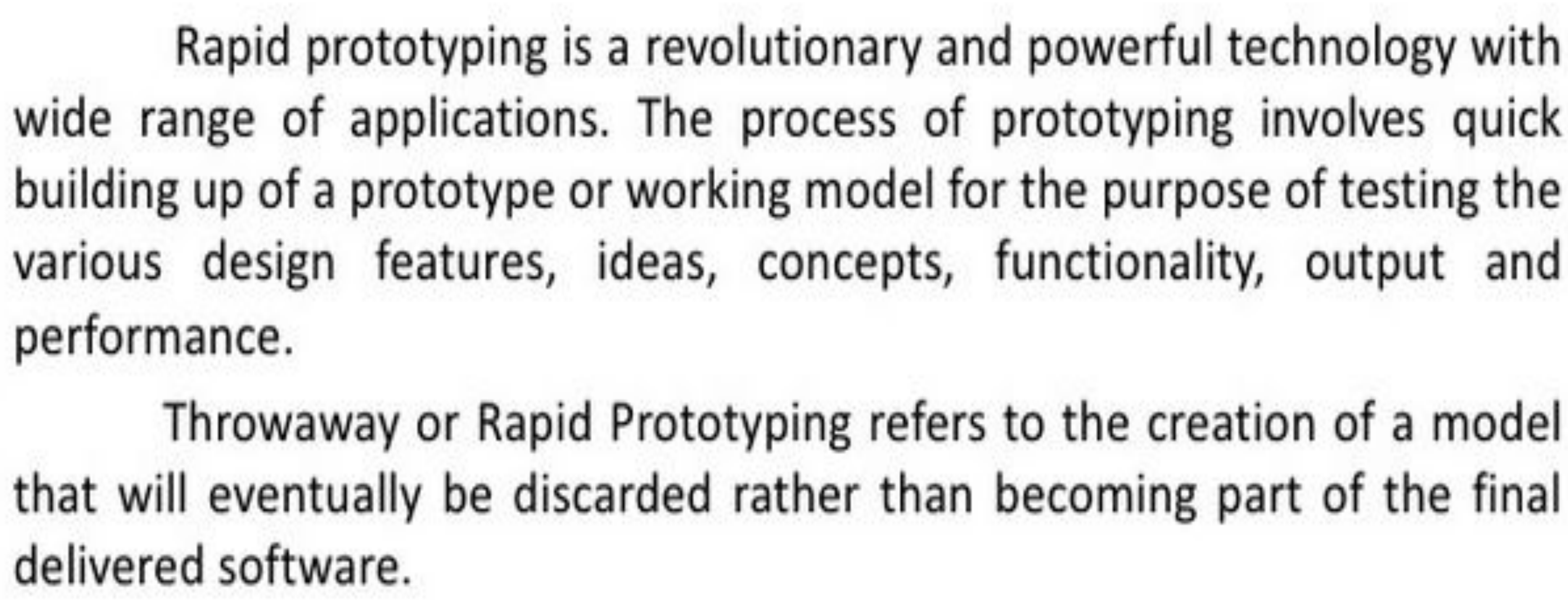
- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.

Rapid Prototyping Model

Customers are non-technical and usually don't know what they want/can have.



Rapid Prototyping Model



Rapid prototyping is a revolutionary and powerful technology with wide range of applications. The process of prototyping involves quick building up of a prototype or working model for the purpose of testing the various design features, ideas, concepts, functionality, output and performance.

Throwaway or Rapid Prototyping refers to the creation of a model that will eventually be discarded rather than becoming part of the final delivered software.

Rapid Prototyping Model

- Developers build a prototype during the requirements phase
- Prototype is evaluated by end users
- Users give corrective feedback
- Developers further refine the prototype
- When the user is satisfied, the prototype code is brought up to the standards needed for a final product.

Rapid Prototyping Model

- A preliminary project plan is developed
- An partial high-level paper model is created
- The model is source for a partial requirements specification
- A prototype is built with basic and critical attributes
- The designer builds
  - the database
  - user interface
  - algorithmic functions
- The designer demonstrates the prototype, the user evaluates for problems and suggests improvements.
- This loop continues until the user is satisfied

Rapid Prototyping Model Strengths

- Customers can “see” the system requirements as they are being gathered
- A more accurate end product
- Unexpected requirements accommodated
- Allows for flexible design and development
- Steady, visible signs of progress produced
- Interaction with the prototype stimulates awareness of additional needed functionality

Rapid Prototyping Model Weaknesses

- Bad reputation for “quick-and-dirty” methods
- Overall maintainability may be overlooked
- The customer may want the prototype delivered.
- Process may continue forever (scope creep)

Rapid Prototyping Model Weaknesses

1. An unstable/badly implemented prototype often becomes the final product.
2. Requires extensive customer collaboration
   - Costs customers money
   - Needs committed customers
   - Difficult to finish if customer withdraws
   - May be too customer specific, no broad market
   - Difficult to know how long project will last
4. Easy to fall back into code-and-fix without proper requirements analysis, design, customer evaluation and feedback.

Use Rapid Prototyping Model When

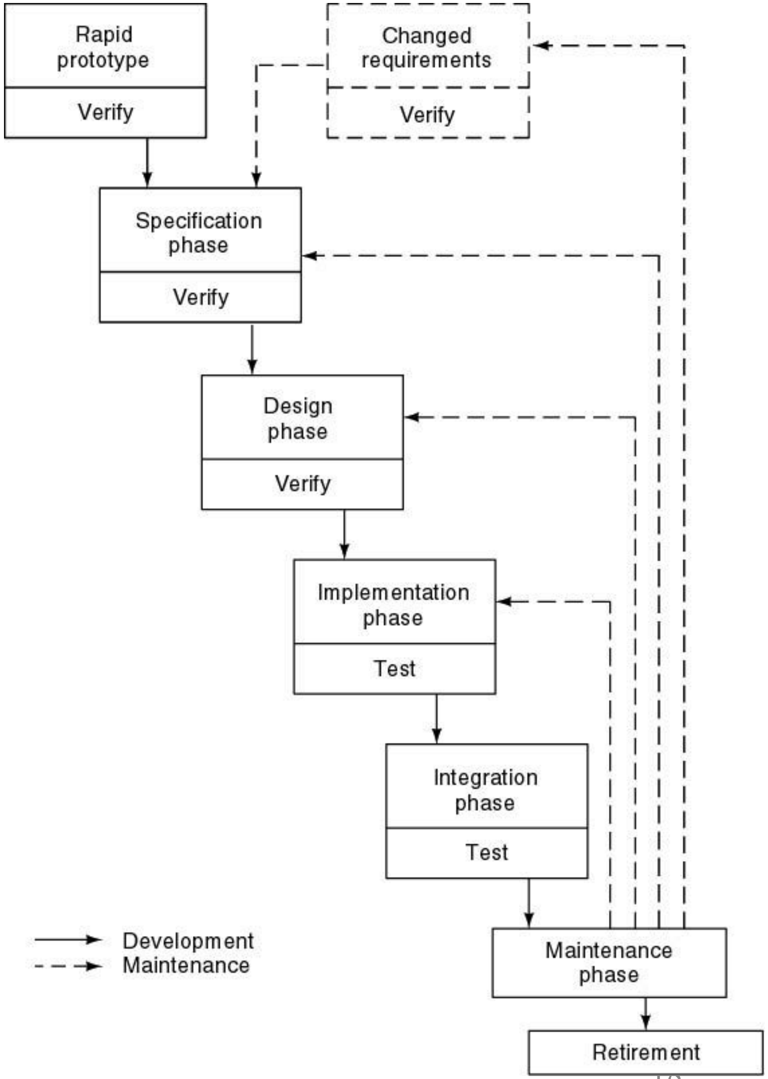
- Requirements are unstable or have to be clarified
- As the requirements clarification stage of a waterfall model
- Develop user interfaces
- Short-lived demonstrations
- New, original development

Key Points

- Rapid prototyping may replace specification phase—never the design phase
- Comparison:
  - Waterfall model—try to get it right first time
  - Rapid prototyping—frequent change, then discard

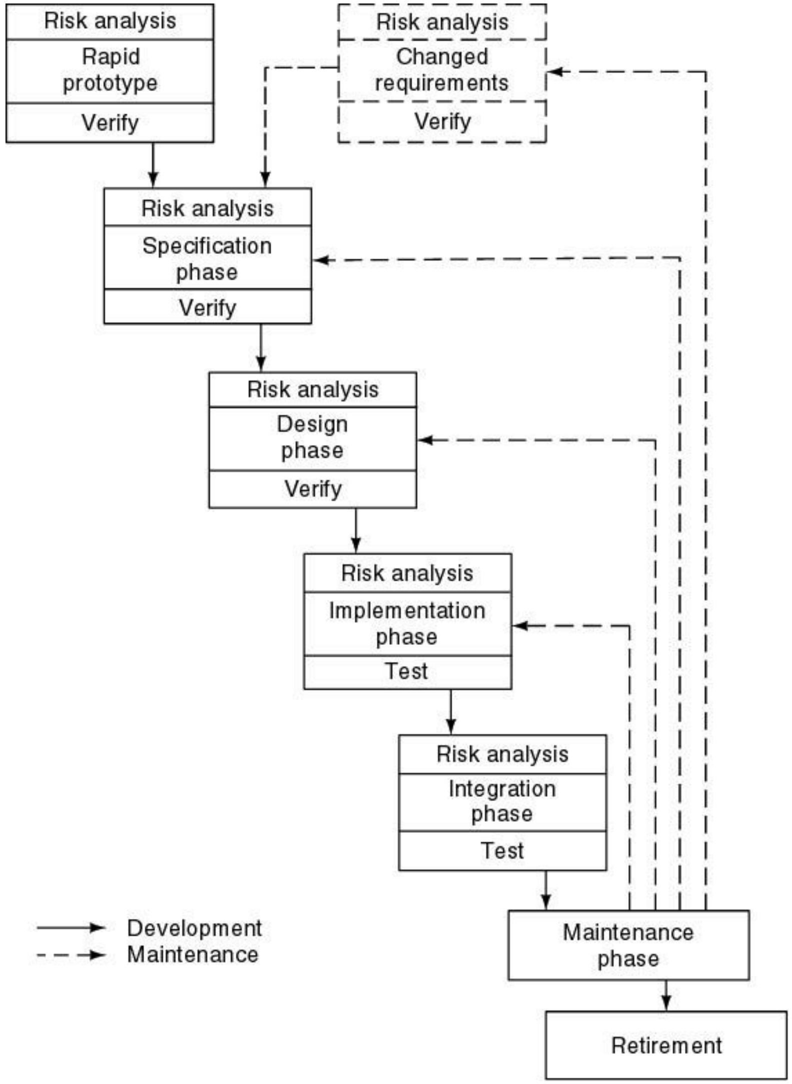
Waterfall and Rapid Prototyping Models

- Waterfall model
  - Many successes
  - Client needs
- Rapid prototyping model
  - Not proved
  - Has own problems



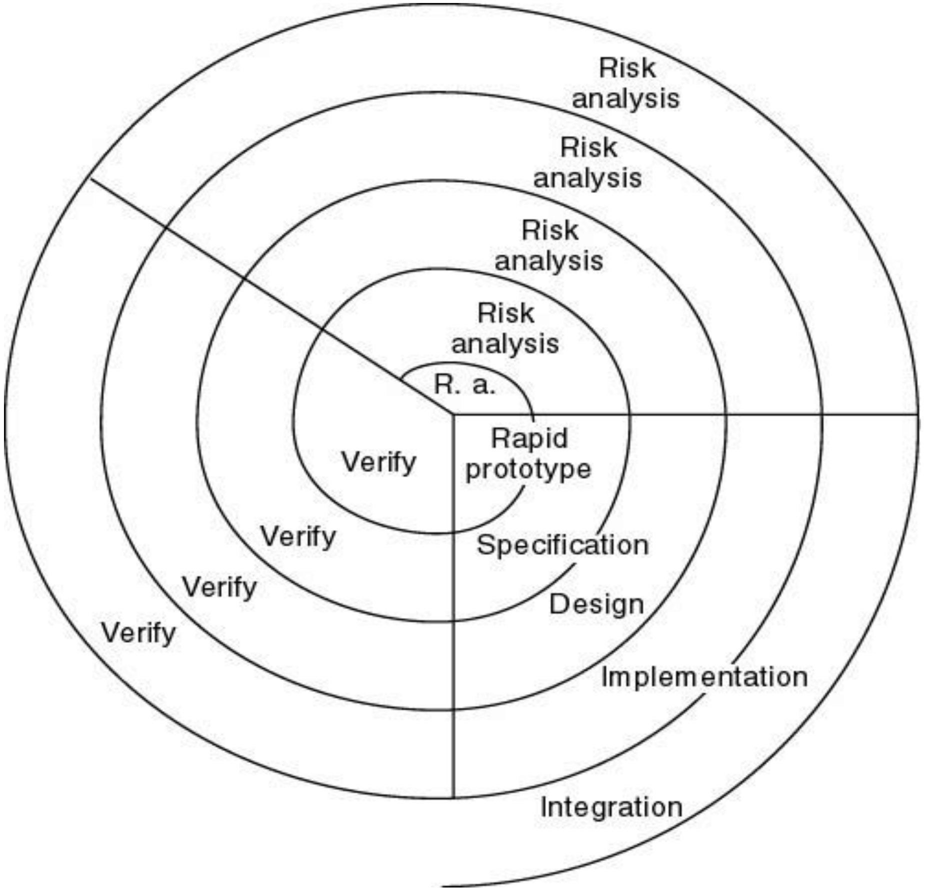
Evolutionary Process Model: Spiral Model

- Simplified form
  - Waterfall model plus risk analysis
- Precede each phase by
  - Alternatives (build, reuse, buy, sub-contract)
  - Risk analysis (lack of experience, new technology, tight schedules)
- Follow each phase by
  - Evaluation
  - Planning of next phase



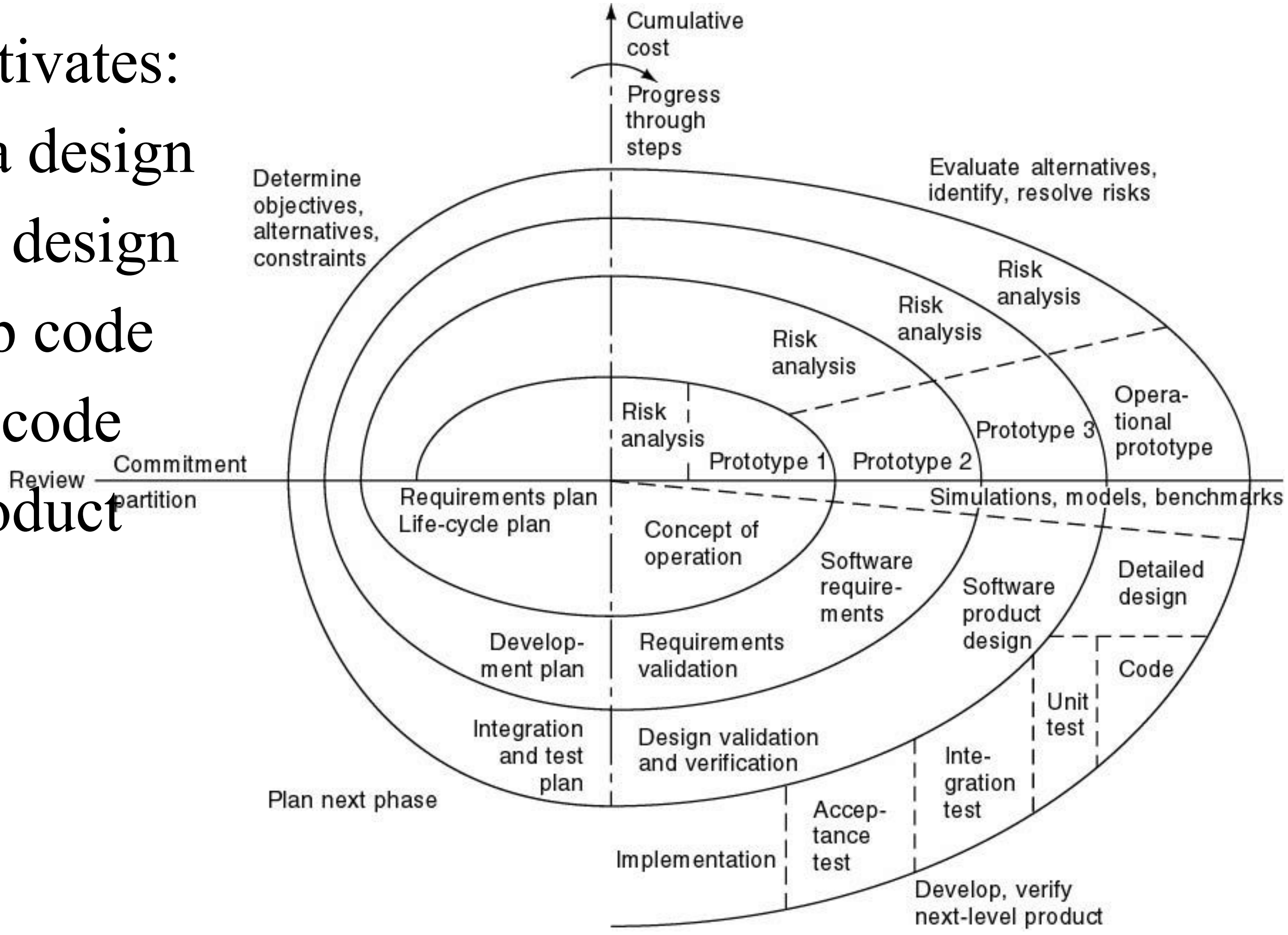
Simplified Spiral Model

- If risks cannot be resolved, project is immediately terminated



Full Spiral Model

- Typical activates:
  - Create a design
  - Review design
  - Develop code
  - Inspect code
  - Test product



Spiral Model Strengths

- Users see the system early because of rapid prototyping tools
- Critical high-risk functions are developed first
- The design does not have to be perfect
- Users can be closely tied to all lifecycle steps
- Early and frequent feedback from users
- Cumulative costs assessed frequently

Spiral Model Weaknesses

- Time spent for evaluating risks too large for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- The model is complex
- Risk assessment expertise is required
- Spiral may continue indefinitely
- Developers must be reassigned during non-development phase activities

Spiral Model Weaknesses

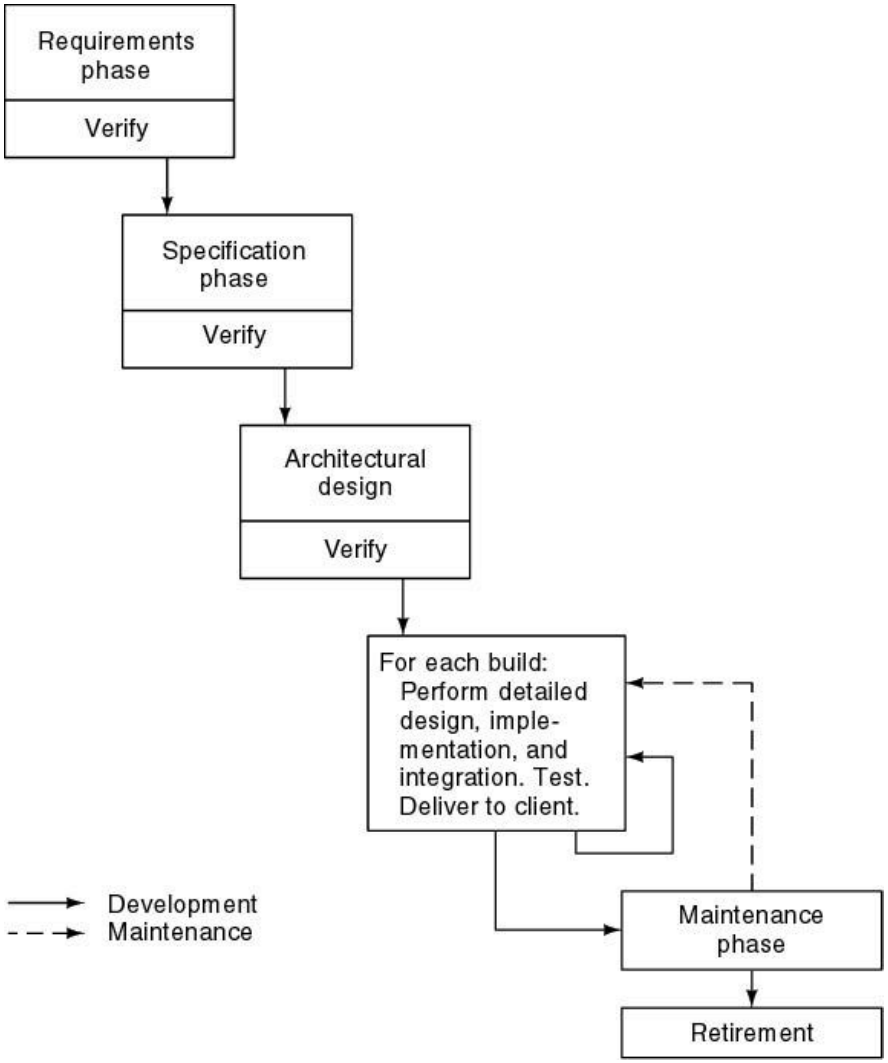
- For large-scale software only
- For internal (in-house) software only

Use Spiral Model When

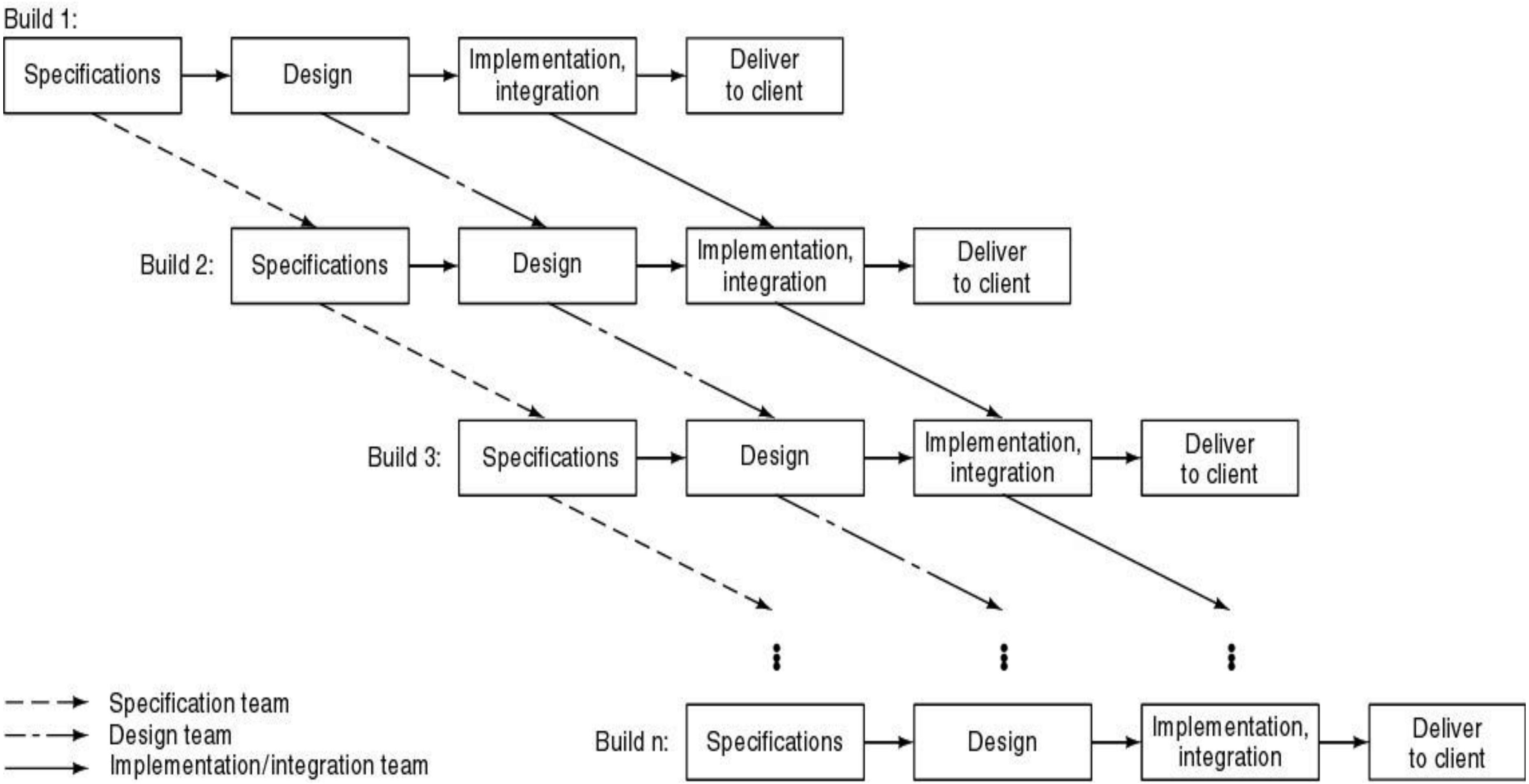
- When costs and risk evaluation is important
- For medium to high-risk projects
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

Incremental Model

- Divide project into *builds*
- Construct a partial implementation of a total system
- Then slowly add increased functionality
- The incremental model prioritizes requirements of the system and then implements them in groups.
- Each subsequent release of the system adds function to the previous release, until all designed functionality has been implemented.



Incremental Model



Incremental Model

- Product at each cycle
- Need open architecture—maintenance implications

Incremental Model

- Waterfall, rapid prototyping models
  - Operational quality complete product at end
- Incremental model
  - Operational quality portion of product within weeks
- Smaller capital outlay, rapid return on investment

Incremental Model Strengths

- Develop major functions first
- Each release delivers an operational product
- Customer can respond to each build
- Uses “divide and conquer” breakdown of tasks
- Lowers initial delivery cost
- Initial product delivery is faster
- Customers get important functionality early
- Risk of changing requirements is reduced

Incremental Model Weaknesses

- Requires good planning and design
- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Well-defined module interfaces are required
- Total cost of the complete system is not lower

Use Incremental Model When

- Funding, schedule, program complexity, or need for early realization of benefits.
- Most of the requirements are known up-front but are expected to evolve over time
- A need to get basic functionality to the market early
- On projects which have lengthy development schedules
- On a project with new technology
- When requirements are defined precisely and no confusion on final product functionality.
- Delivers quick product with limited functionality.

Conclusions

- Different life-cycle models
- Each with own strengths
- Each with own weaknesses
- Criteria for deciding on a model include
  - The organization
  - Its management
  - Skills of the employees
  - The nature of the product
- Best suggestion
  - “Mix-and-match” life-cycle model

Home Work

- What is software engineering? How is it different from other traditional engineering branches?
- Distinguish between a software product and a software process.
- What are the characteristics of software product?
- Name two or three applications that would be more difficult to prototype.
- Explain why a software system that is used in real world environment must change or become progressively less useful.

Home Work

- What do you understand by software development lifecycle in SDLC? Why it is important to adhere to a life cycle model while developing a large software product?
- Suppose you are working as a software engineer involved in the development of an e-commerce website. What are 2 most important characteristics your software must have?

Home Work

- Discuss the pros And cons of letting people rotate between projects of different application domains
- Why software quality organization is independent of the development organization
- Explain how both the waterfall and the prototyping model of the software process can be accommodated in the spiral process model?
- What is the importance of models in software engineering? Explain with examples of any three process models which are commonly used.

Home Work

- What are the potential advantages of adhering to life cycle models for software?
- Extreme programming express user requirements as stories, with each story written on a card. List and explain at least two respectively, advantages and disadvantages of this approach to requirements description.
- Apart from the challenge of heterogeneity, rapid delivery and trust, identify at least three problems and challenges that Software Engineering is likely to face in the 21st century.

Home Work

- List at least two advantages and two disadvantages professional software engineer should be certified in the same way as doctors or lawyers.
- Distinguish between process, methods and tools.
- Describe four professional responsibilities of software engineer
- What is process iteration, explain spiral model
- What are characteristics of rapid software development?

Home Work

- Statement of work
- List of stakeholders