The Software Development Life Cycle

Review

- What is the software process?
  - the ***activities***, techniques, tools, and individuals to produce software
- Documentation important for each phase. Why?
  - people & products change
- Testing is essential throughout life cycle
  - traceability?
    - tying a piece of doc/design back to previous doc
- Why are there no "silver bullets"?
  - software is inherently hard because of complexity, conformity, changeability, invisibility

Software Life-Cycle Model

- Definition
  - The series of steps through which the product progresses
- The models specifies
  - the various phases of the process
    - e.g., requirements, specification, design…
  - the order in which they are carried out

Variations in the Process

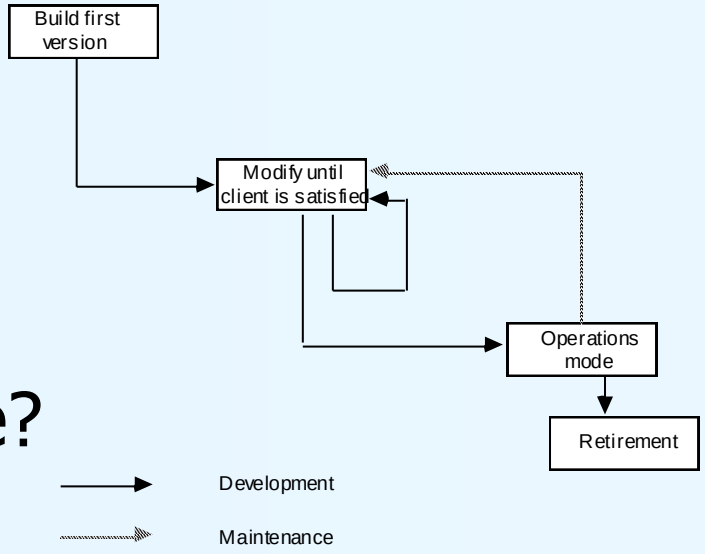
- Regardless of exact procedure, all broadly follow the seven phases of SW Life Cycle
  - Requirements phase
  - Specification phase
  - Design phase
  - Implementation phase
  - Integration phase
  - Maintenance phase
  - Retirement
- Some use different terms – some combine phases

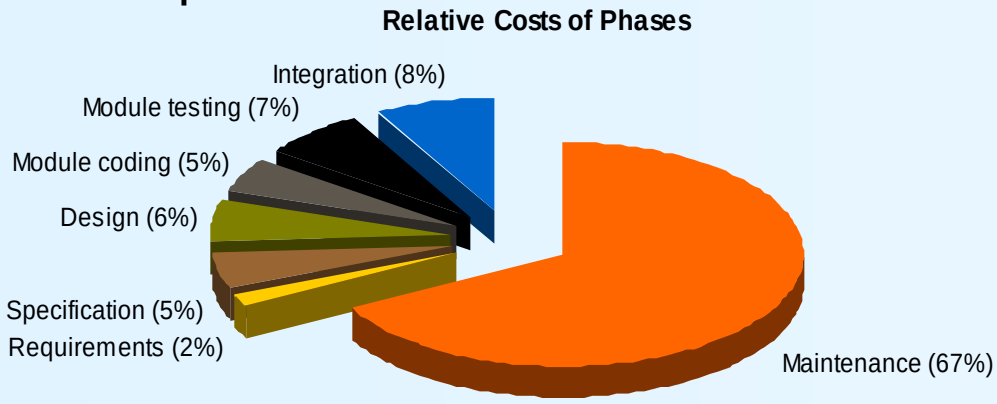
Importance of Lifecycle Models

- Provide guidance for project management
  - what major tasks should be tackled next? milestones!
  - what kind of progress has been made?
- The necessity of lifecycle models
  - character of software development has changed
    - early days: programmers were the primary users
    - modest designs; potential of software unknown
  - more complex systems attempted
    - more features, more sophistication → greater complexity, more chances for error
    - heterogeneous users

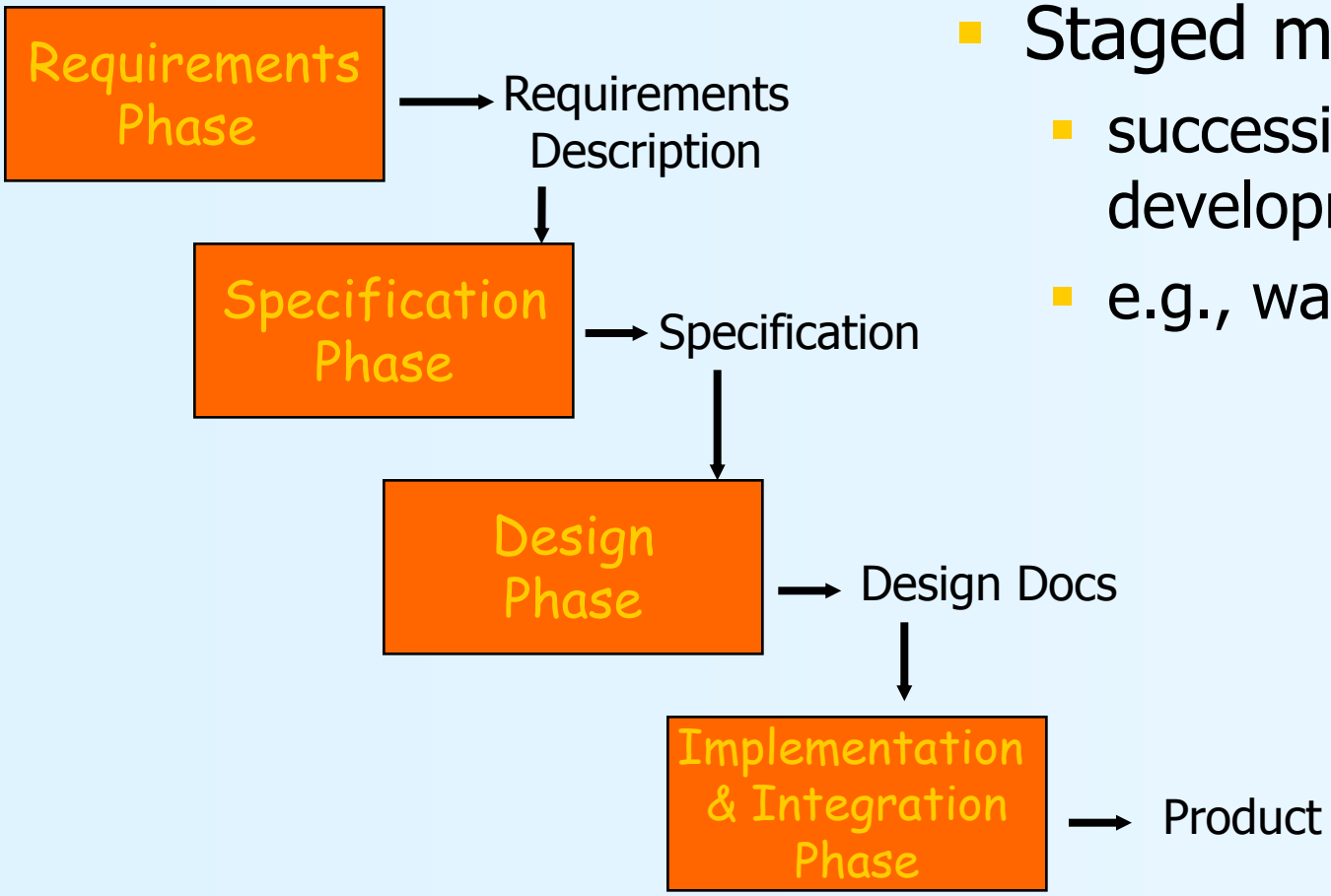
Lifecycle Models

- Build-and-fix
  - develop system
    - without specs or design
    - modify until customer is satisfied
- Why doesn't build-and-fix scale?
  - changes during maintenance
    - most expensive!





Lifecycle Models



- Staged models
  - successive stages of development
  - e.g., waterfall model

Lifecycle Models

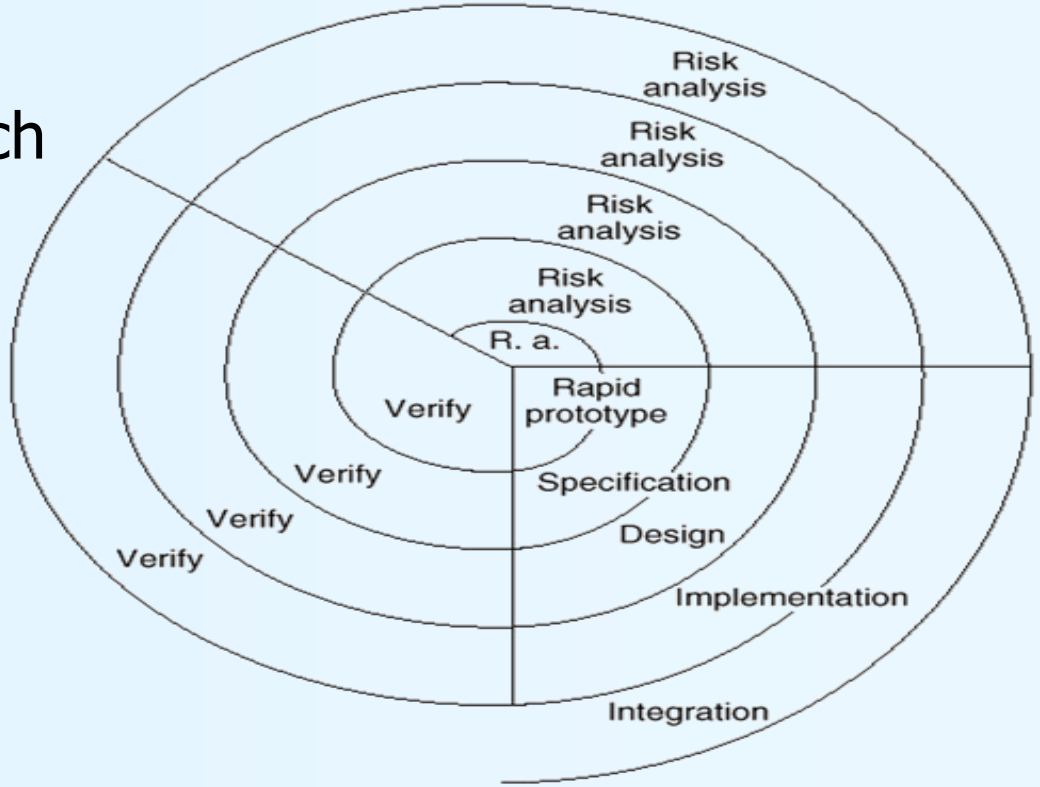
- Rapid prototyping
  - build something users can understand & assess
  - often focuses on the interface
  - waterfall model follows the prototype
- Incremental (aka Evolutionary) development
  - incrementally expand the system
  - can be used to do a "phased delivery" to users
    - more expectation management needed!
    - build-and-fix revisited?

Lifecycle Models

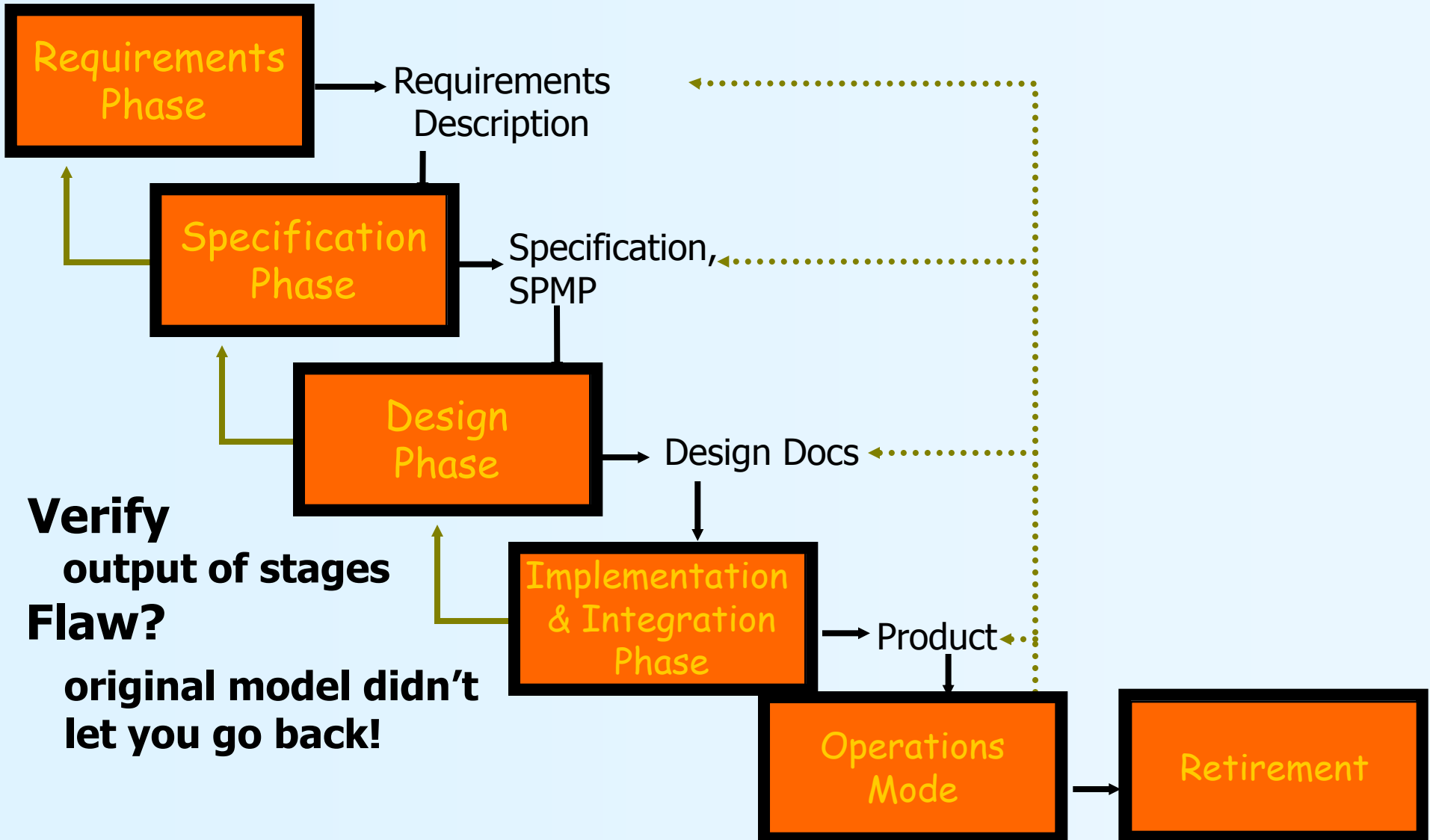
- Spiral Model
  - risk-driven approach
  - assess risks before each phase
  - re-assess in frequent cycles

Figure 3.7
Portion of Figure 3.6 redrawn as a spiral.

TM-14



Waterfall Model



Verify

output of stages

Flaw?

original model didn't let you go back!

Advantages of Waterfall Model

- Enforced discipline through documents
  - no phase is complete until the docs are done & checked by SQA group
  - concrete evidence of progress
- Testing is inherent in every phase
  - continuously as well as at end of phases

Drawbacks of Waterfall Model

- Document-driven model
  - customers cannot understand these
  - first time client sees a working product is after it has been coded.
    - leads to products that **don't meet customers needs**
- Assumes feasibility before implementation
  - re-design is problematic
  - works best when you know what you're doing
    - when requirements are stable & problem is well-known

Rapid Prototyping

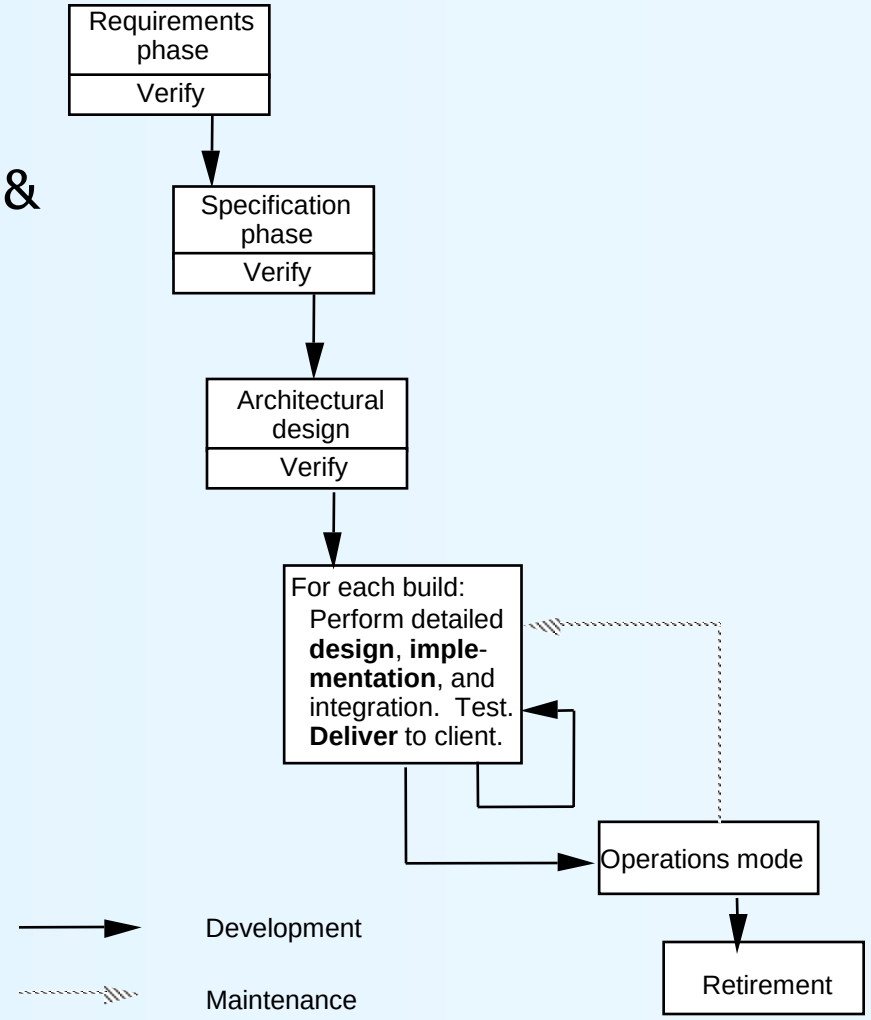
- Rapid prototyping as a requirements tool
  - allow users to "see" & use proposed solutions
  - develop specification from the prototype/requirements
    - proceed with rest of stages in waterfall model
- Prototype must be constructed & changed quickly
  - do not spend a lot of time perfecting the code/structure
    - plan to throw it away
    - put in front of customer ASAP
  - user interface prototyping & other rapid development tools make this easier

Assessment of RP Model

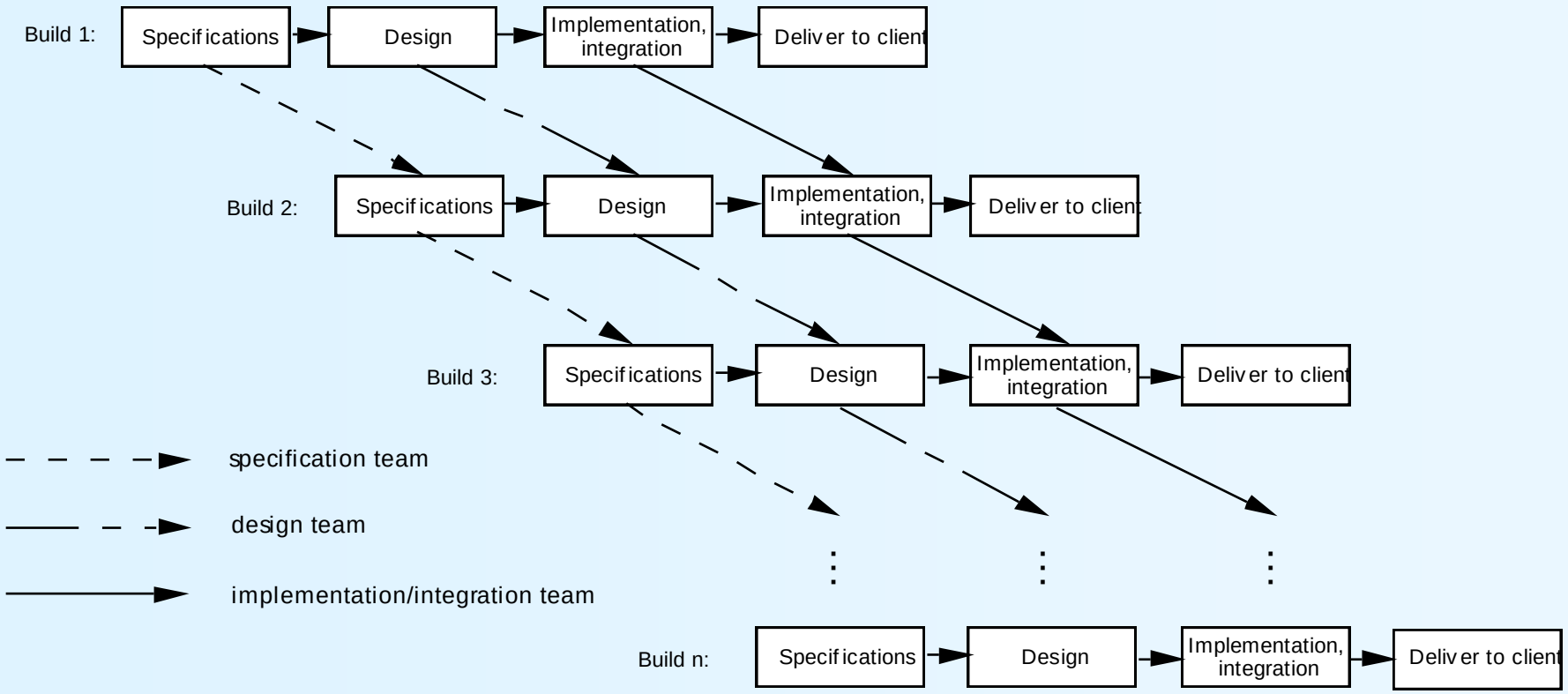
- Advantages
  - process proceeds linearly (less need for feedback loops)
  - easier to take technology risks with the prototype
- Disadvantages
  - expectation management
  - turning prototypes into production code

Incremental Model

- Divide project into *builds*
  - each adds new functions
  - each build integrated w/ structure & product tested as a whole
- Advantages
  - operation product in weeks
  - less traumatic to organization
  - smaller capital outlay, rapid ROI
- Disadvantages
  - need an open architecture
  - too few builds → build-and-fix
  - too many builds → overhead



Other Incremental Models

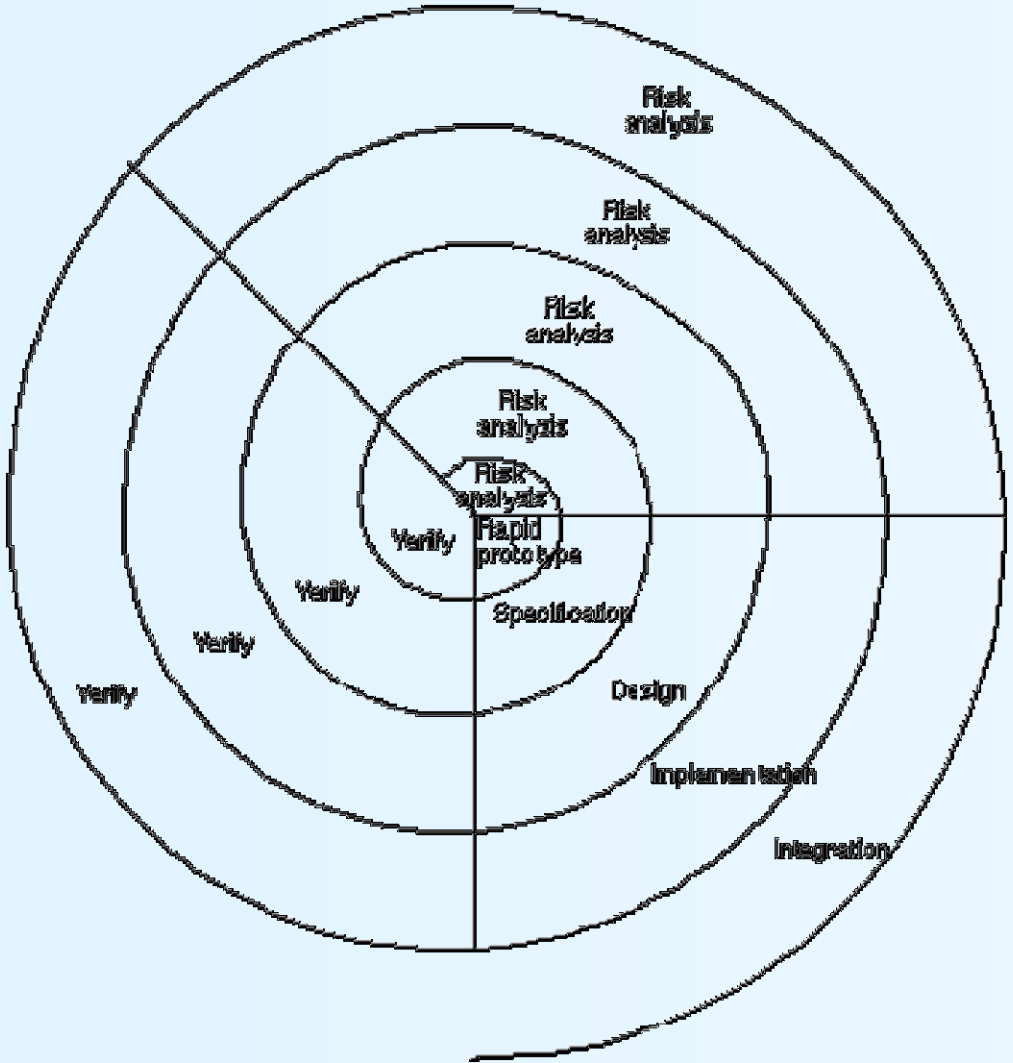


- Advantages
  - more parallelism saves lots of time!
- Risks
  - no overall design at start → pieces might not fit together

Spiral Model

- Always some risk involved in software development
  - people leave… other products not delivered on time…
- Key idea
  - minimize risk
    - e.g., building prototypes & simulations minimizes risks
- Precede each phase by
  - looking at alternatives
  - risk analysis
- Follow each phase by
  - evaluation
  - planning of next phase

Spiral Model



Risk Assessment

- Risk-driven approach in each spiral:
  - identify potential risks
  - plan next step based on risk analysis
  - refine design in highest-risk areas
- Explicitly attempts to identify potential problems
  - not just in initial stages of design
  - also later, when more has been learned about the problem and the design
- What are the "risky" parts of the system
  - relies on developer experience

Assessing the Spiral Model

- Risk assessment
  - how does one do it?
  - how is it known that "proper" risks have been identified?
  - "experience" is a critical factor
  - model fails if risks are inaccurately assessed
- Not clear it works for contract projects
  - how do you cancel a contract in the middle?
- Mainly for very large projects
  - risk assessment could cost more than development!

Dangers & Promises of Lifecycles

- Managers love them
  - defines a set of "deliverables"
  - can tell upper management that "stage so-and-so is completed"
- Programmers find them inadequate
  - customers can't adequately state requirements up-front
  - stages become intermixed with others
    - e.g., design reveals that part of specification can't be cost-effectively implemented