CISC/CMPE 327 Software Quality Assurance

Queen's University, 2019–fall

Lecture #2
Software Process Models

Software Process Models

- Quality in Context
  - To understand the roles of quality assurance in software development, we must understand how software development works
    - We cannot discuss inspection, testing, and metrics in a vacuum
  - As background, therefore, we will begin by reviewing:
    - Major process models of the software development community, the ways software development efforts are organized
    - Some ways of assessing development process quality
    - Quality management standards for software processes

But first… Why bother?

- Example: U.S. Federal Aviation Administration
  - Operating an archaic air traffic control system
  - Started examining replacement options in 1981
  - By 1994, project was shelved after a cost of more than $2.6 billion and a lapsing delivery date
  - Some estimates put the economic cost of flight delays at $50 billion per year
  - Software is hard to replace!
    - Getting it right the first time is important

http://spectrum.ieee.org/computing/software/why-software-fails

Software Process Models

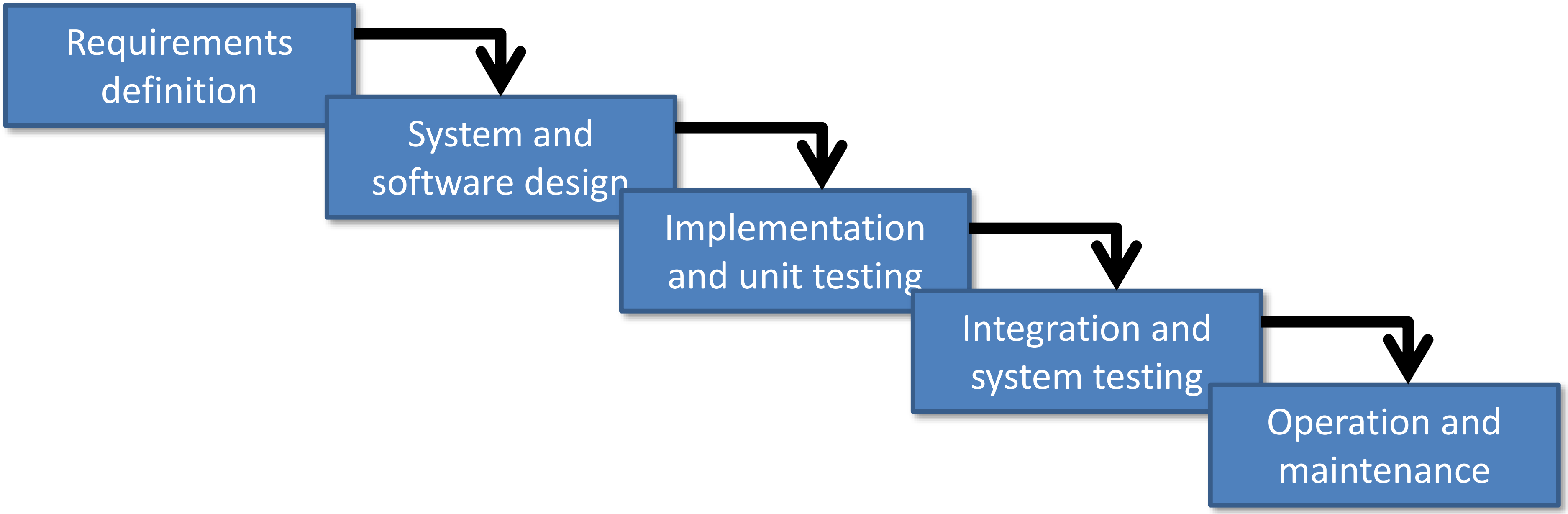
- Software Process Models
  - A software development process is a method for developing computer software that organizes the effort into a number of separate tasks and steps
  - This helps make it possible to develop large software systems using many people in an organized, manageable, and trackable way to retain control of the development
  - Having control addresses QA principle 1: know what you are doing

Software Process Models

- Fundamental Process Activities
  - All software process models share four fundamental process activities and differ primarily in how they are organized and interleaved
    - Specification: define requirements, functionality, and constraints
    - Development: build software to meet the specification
    - Validation: validate that it does what the customer wants
    - Evolution: evolve to meet changing needs and expectations

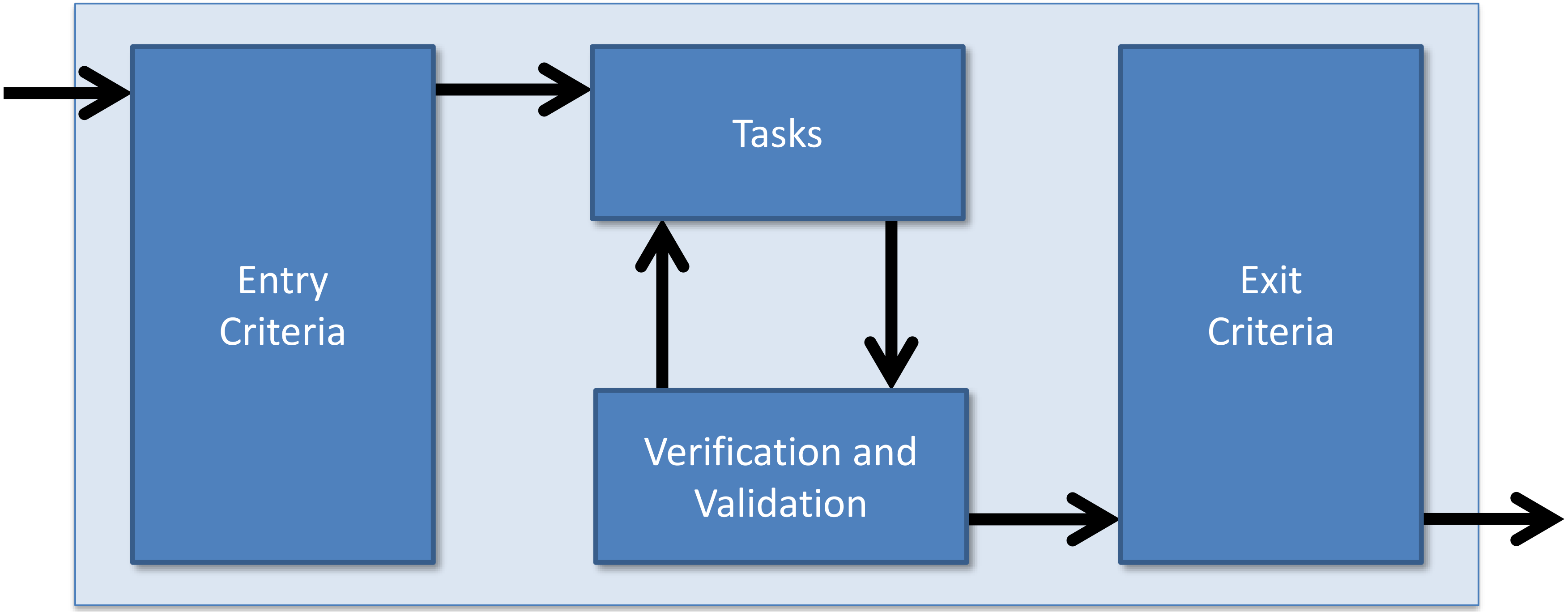
The Waterfall Model

- Original Waterfall Model
  - First explicit model, derived from other engineering processes
  - Cascade of phases, carried out in order, with sign-off of each before proceeding to the next


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

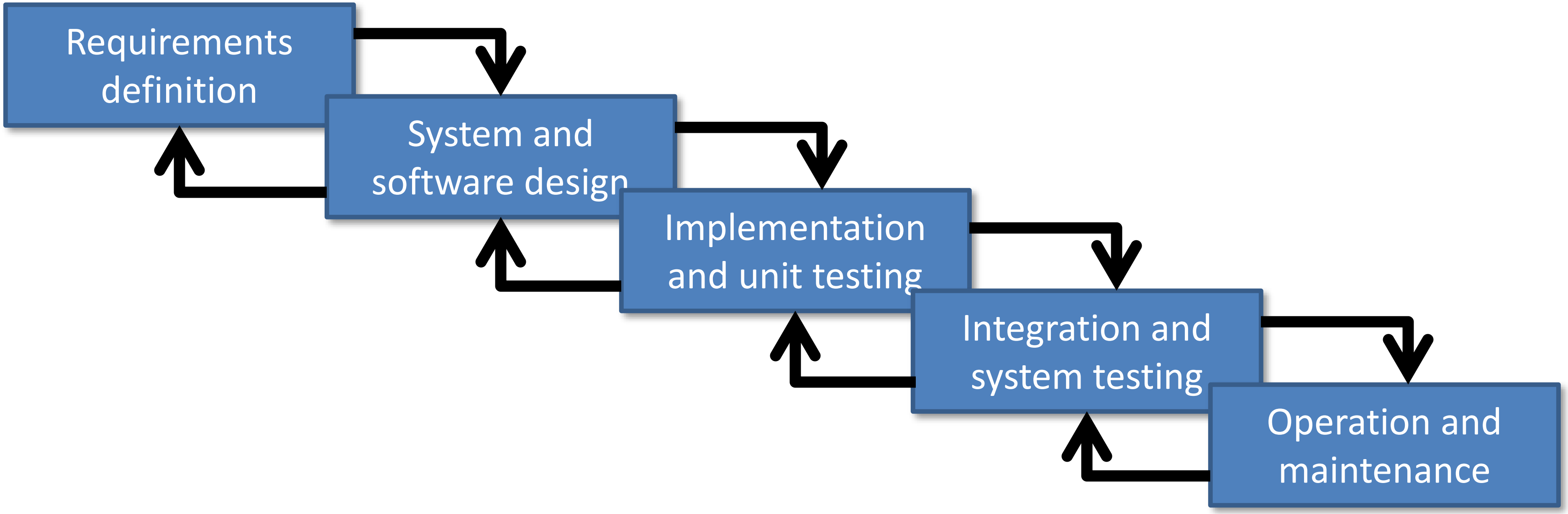
The Waterfall Model

- Organizes quality control
  - IBM's ETVX - Entry, Task, Validation, eXit at each step



The Waterfall Model

- Iterative Waterfall Model
  - Refined to be more realistic with practice
  - Go back up waterfall to revisit previous steps as necessary
  - Still work on one step at a time, cascade to next as completed



Requirements definition

System and software design

Implementation and unit testing

Integration and system testing

Operation and maintenance

The Waterfall Model

- 1. Requirements Analysis and Definition
  - System's required services, constraints, and goals are established by consultation with users/customers
  - Expressed in a way understood and agreed to by both users and developers
    - often test cases or scenarios
  - Quality control
    - requirements reviews (inspection)

The Waterfall Model

- 2. System and Software Design
  - Partitions into hardware and software subsystems
  - Establishes overall system and software architecture
  - Establishes functional specifications for components of the architecture
  - Quality control
    - Design reviews (inspection)

The Waterfall Model

- 3. Implementation and Unit Testing
  - Design realized as a set of programs and program components (units) to implement components of the architecture
  - Verify that units meet functional specifications
  - Quality control
    - Unit testing, component testing

The Waterfall Model

- 4. Integration and System Testing
  - Integrate individual programs and program units into complete system
  - Validate system that system meets requirements
  - Quality control
    - Integration testing, acceptance testing

The Waterfall Model

- 5. Operation and Maintenance
  - Normally longest phase of software life cycle
  - Install system and put into use
  - Maintenance involves correcting errors discovered in practice (“failures") and improving system units (e.g., performance tuning) and enhancing services in response to new requirements
  - Quality control
    - Regression testing, acceptance testing

The Waterfall Model

- 6. Retirement and Decommissioning
  - System is retired and replaced with a new one
  - Rarely done now because of cost and risk of replacement
    - Continuous evolution more common

Drawbacks of the Waterfall Model

- Early Freezing
  - In practice, frequent iterations back up the waterfall make it difficult to identify checkpoints and track progress
  - Therefore it is normal to freeze parts of the development, such as requirements and design, and move on to the later stages quite early without feedback

Drawbacks of the Waterfall Model

- Early Freezing
  - Premature freezing of requirements may mean that the system won't end up doing exactly what the users want
  - Premature freezing of designs often leads to badly structured systems as design problems are worked around using implementation tricks

Drawbacks of the Waterfall Model

- Inflexible Partitioning
  - The inflexible partitioning into distinct stages, while a management advantage, often leads to undesirable technical results
  - Delivered systems are sometimes unusable, do not meet users' real requirements (as opposed to their original guesses)

Drawbacks of the Waterfall Model

- But…
  - The waterfall model reflects common engineering practice
  - Likely that this process model will still remain the norm for some time

The Prototyping Model

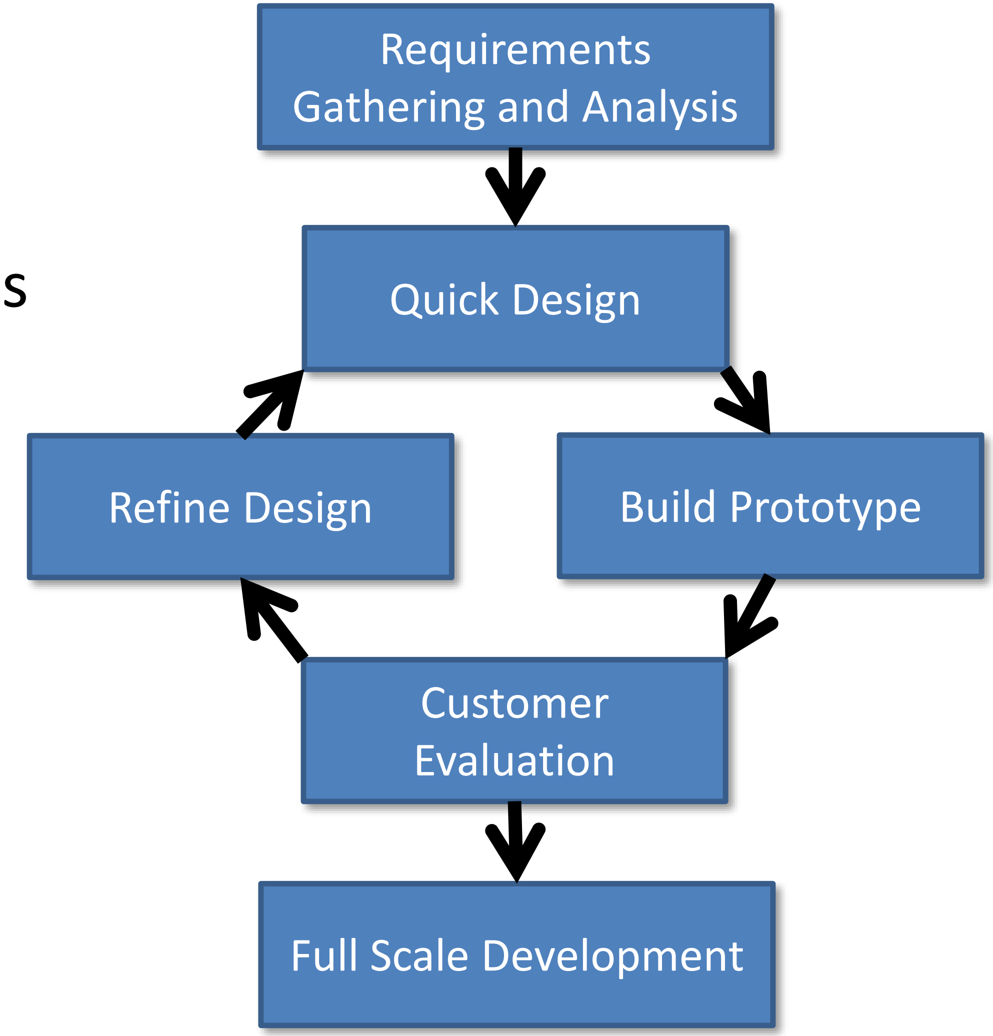
- Problems with Requirements
  - First step in the waterfall is requirements gathering and analysis
  - In practice, this is the most difficult part, and experience with the waterfall indicates that most failures are due to inadequate requirements understanding
  - Users often change requirements as they see what can be done

The Prototyping Model

- Prototyping
  - The prototyping model attempts to address the requirements difficulty by introducing an iterative, by example requirements stage
  - A prototype is a partial implementation of a software system with all external interfaces presented
  - Users use the prototype and provide feedback from which real requirements are gradually refined
  - Final prototype serves as example of intended system

The Prototyping Model

- Prototyping Model
  - Extend requirements phase to include a sequence of prototypes
  - Improve requirements and design as prototypes refined
  - When users and developers are both satisfied, move on to real development



The Prototyping Model

- 1. Requirements Gathering and Analysis
  - Much like waterfall model, but less stringent since prototype will help expose inadequacies
  - Quality control
    - Requirements reviews (inspection)
- 2. Quick Design
  - Make a simple approximate initial design, refine during prototype iteration
  - Quality control
    - Prototype testing

The Prototyping Model

- 3. Build Prototype
  - Quickly hack together an approximate implementation showing salient external features
  - Quality control
    - Essentially none
- 4. Customer Evaluation
  - Users validate prototype, report inadequacies
  - Quality control
    - Acceptance testing and evaluation (inspection)

The Prototyping Model

- 5. Design Refinement
  - Refine design in response to user feedback from prototype
  - Quality control
    - Design reviews (inspection)
- 6. Full Scale Development
  - Remaining stages of traditional waterfall model

Drawbacks of Prototyping Model

- Wasted Work
  - Prototypes are normally built using substandard quality controls (“thrown together”) to speed the iteration (“quick turnaround”)
  - Thus they must be discarded after the prototyping phase, even if they solve significant problems

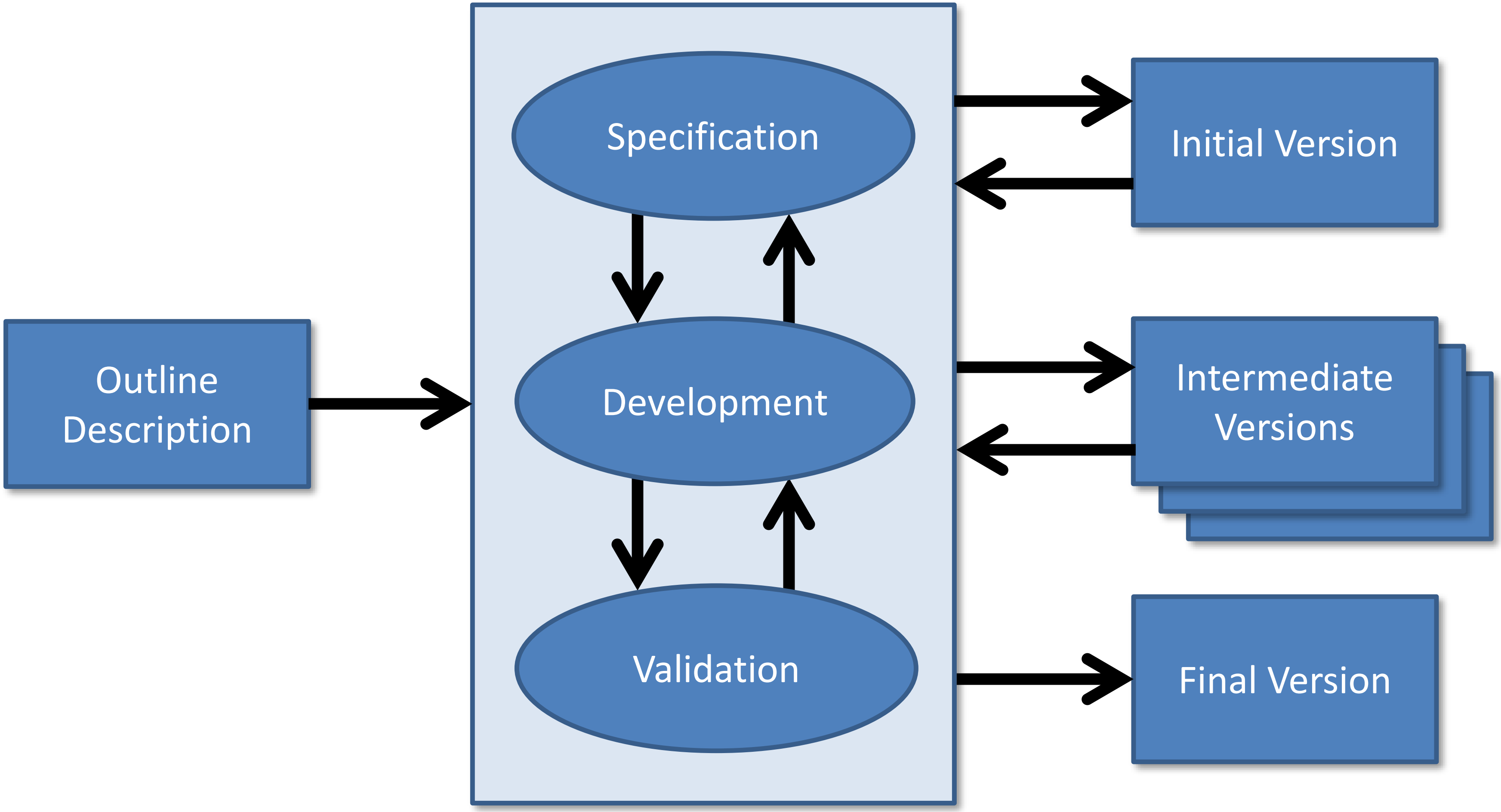
Drawbacks of Prototyping Model

- Inadequate or Incomplete Prototypes
  - Full prototypes of complex systems can be difficult or impossible to create quickly
  - Thus prototypes are often done in parts, which may miss critical requirements at the integration or complete system stage
- When to Stop Iterating
  - Easy to have users convince you to continue refining beyond the point where requirements and design are sufficient ("creeping excellence")

Evolutionary Development

- Prototype Evolution
  - Evolutionary prototyping is a method to avoid wasting work and take advantage of "creeping excellence" by smoothly evolving the initial prototype to the final product
  - In essence, never leave prototype iteration until implementation is complete

Evolutionary Development



Summary

- Software Process, Part I
  - Software development has four tasks
  - Software development processes differ in how these are interlaced
  - Oldest and most common process is the Waterfall Process
  - Some recent and popular processes are based on Prototyping

Summary

- Today's References
  - Kan, Metrics and Models in Software Quality Engineering
    - Ch. 2, Software Development Process Models
  - Sommerville, Software Engineering
    - Ch. 2, Software Processes
- Next time
  - More software process models