SE 2AA4, CS 2ME3 (Introduction to Software Development)

Winter 2018

03 Software Quality (Ch. 2)

Dr. Spencer Smith

Faculty of Engineering, McMaster University

January 10, 2018

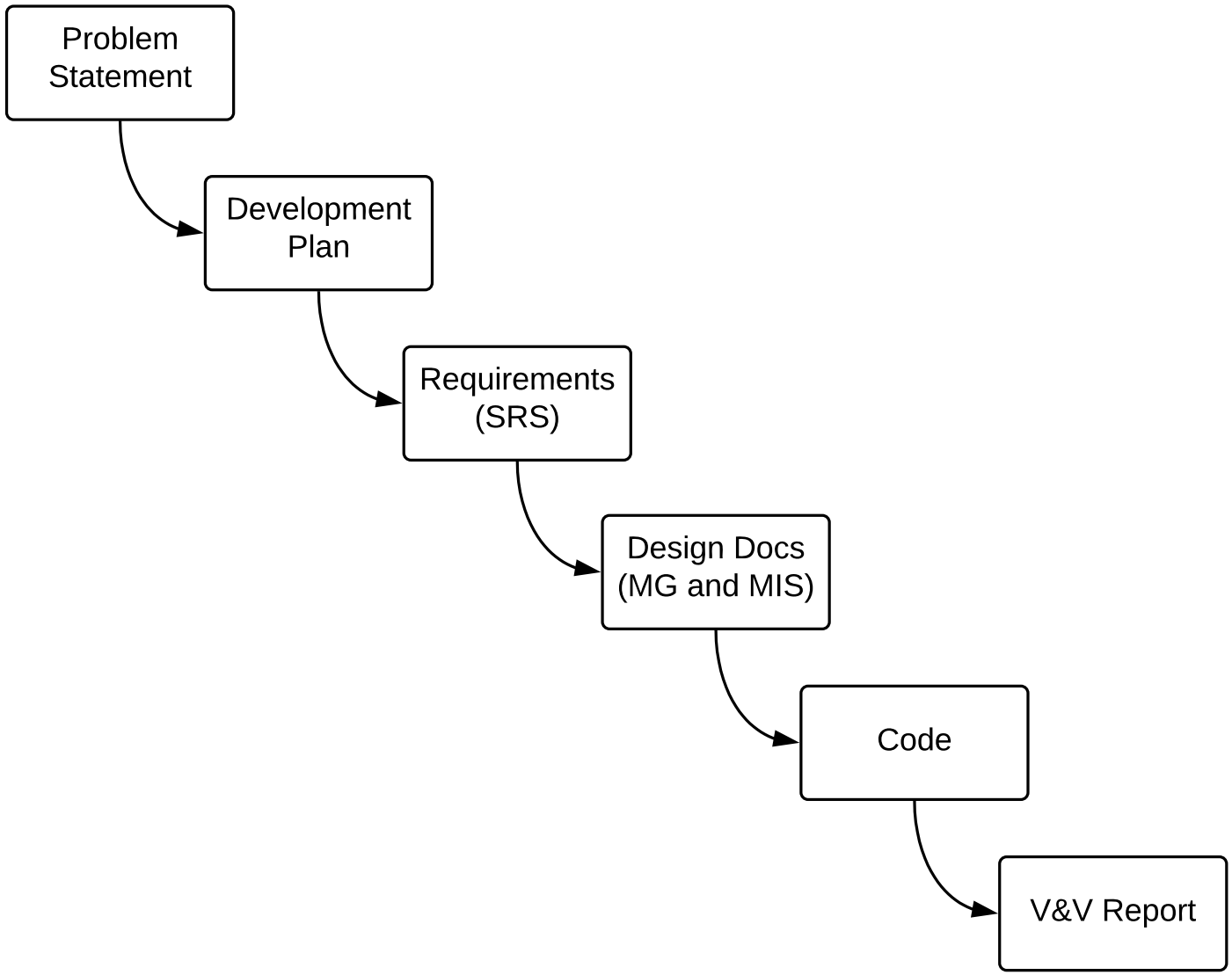
03 Software Quality (Ch. 2)

- Administrative details
- Software development process
  - Software documentation
  - Software development phases
  - Software life cycle models
- How software differs from other engineering products
- Definition of quality
- Sample qualities

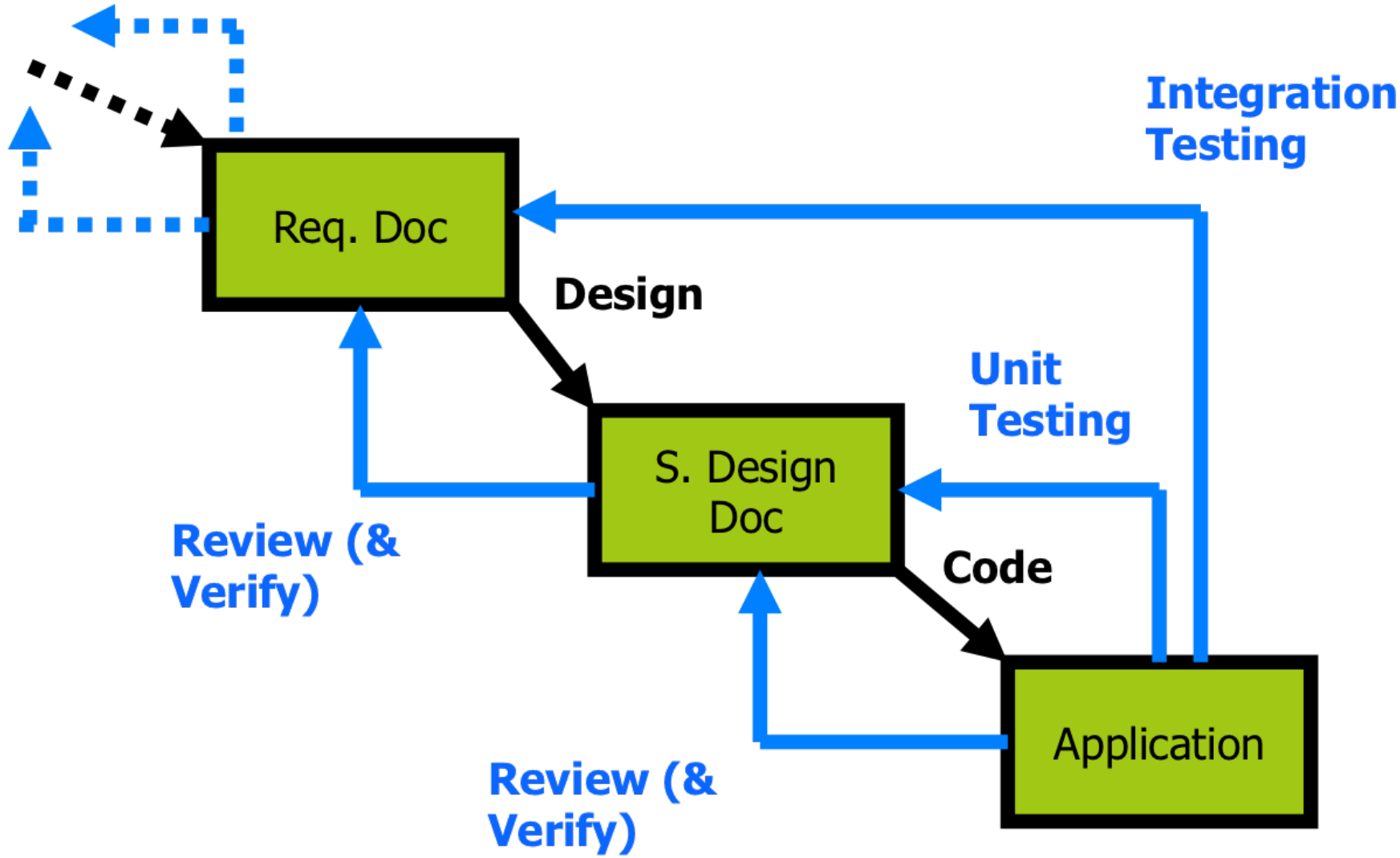
Administrative Details

- Assignment 1
  - Part 1: January 22, 2018
  - Partner Files: January 28, 2018
  - Part 2: January 31, 2018
  - Correction to quadratic interpolation formula
  - Clarification that input data files should also be under version control

Rational Design Process



Software Lifecycle



Software Documentation

- Every software product should include documentation that presents the product to clients, reviewers, users and maintainers
- It is useful to produce documentation that makes it appear as if the software product was developed by a rational process
  - Mathematics have long followed this approach in presenting results
  - See Parnas and Clements, 1986, "A Rational Design Process: How and Why to Fake It,"
  - See Parnas, 2010, "Precise Documentation: The Key to Better Software" in The Future of Software Engineering (2010)

Software Development Process

- A rational development process is needed to produce quality software
- Any proposed rational process is necessarily an idealization
  - Humans inevitably make errors
  - Communication between humans is imperfect
  - Many things are not understood at the start
  - Supporting technology always has limitations
  - Requirements change over time

Software Development Process

1. Requirements: What is the problem that needs to be solved? What are the product requirements that need to be satisfied? (SRS)
2. Design: How will the problem be solved? How will the product requirements be satisfied? (MG, MIS)
3. Implementation: What is a solution to the problem? What is an executable implementation of the design? (Code)
4. Verification: What behaviour does the product exhibit? Is the behaviour correct? (VnV plan, VnV report)
5. Delivery and Maintenance: How will the product be delivered? What needs to be maintained? How will it be maintained?

Software Life Cycle Models

- Waterfall model
  - Development follows the logical order of the phases given above in a linear fashion
  - Is an idealization of the software development process that is rarely realized
  - Potentially appropriate when requirements are well understood and slow to change
- Other life cycle models
  - Refinement
  - Incremental
  - Spiral
  - Prototyping

How Software Differs from other Engineering Products

- Intangible
  - Not physical
  - Hard to visualize
  - Hard to separate what is key from what is incidental
- Malleable
  - Easy to modify
  - But modification requires care
- Human intensive
  - Software production is 99.9% engineering, 0.1% manufacturing
  - Software is essentially documentation

Question

Every software system should be designed to achieve an uptime of 24 hours a day, 7 days a week, 365 days a year.

A. True

B. False

What is Quality?

- Definition of quality?
- Quality of a McDonald's hamburger versus steak?
- Quality of BMW versus Ford Escort?
- Beta versus VHS?
- Blu Ray versus HD DVD?
- Mac OS X versus Linux versus Windows?

What are the Important Qualities for Software?

- Brainstorm

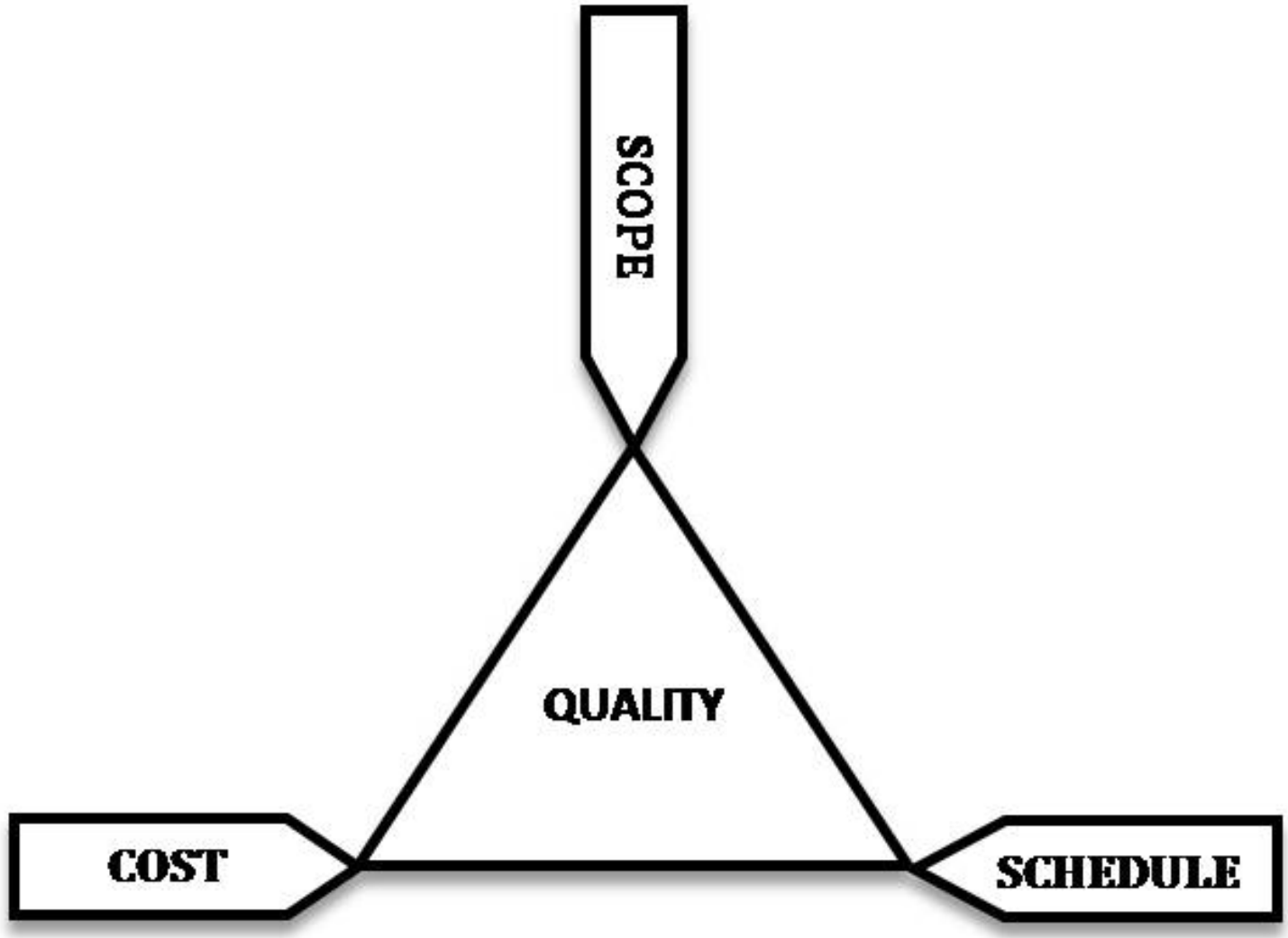
Definition of Software Qualities

- Measures of the excellence or worth of a software product (code or document) or process with respect to some aspect
- Aspects include
  - correctness
  - reliability
  - robustness
  - performance
  - verifiability
  - productivity
  - etc.
- User Satisfaction = The Important Qualities are High + Within Budget

Pick Any Two

Wikipedia Project Management Triangle

Project Management Triangle



Wikipedia Project Management Triangle

Software Qualities

- The goal of software engineering is to produce quality software. But what are the desirable qualities that software should possess?
- External versus internal software qualities
  - External qualities are visible to the user
  - Internal qualities are visible to the developer
  - Internal qualities help external qualities be achieved
- Product versus process qualities
  - Product qualities concern the product itself
  - Process qualities concern how the product is developed
  - Process qualities help product qualities be achieved
  - Process qualities can also reduce development costs
- The importance of a particular software quality varies across software products - external qualities are not as important for embedded systems as for desktop software

Correctness Versus Reliability Versus Robustness

What might be the difference between these 3 qualities?

Can you assess correctness without a requirements specification?

Correctness

- A software product is correct if it satisfies its requirements specification
- Correctness is extremely difficult to achieve because
  - The requirements specification may be imprecise, ambiguous, inconsistent, based on incorrect knowledge, or nonexistent
  - Requirements often compete with each other
  - It is virtually impossible to produce "bug-free" software
  - It is very difficult to verify or measure correctness
- If the requirements specification is formal, correctness can in theory and possibly in practise be
  - Mathematically defined
  - Proven by mathematical proof
  - Disproven by counterexample

Reliability

- A software product is reliable if it usually does what is intended to do
- Correctness is an absolute quality, while reliability is a relative quality
- A software product can be both reliable and incorrect
- Reliability can be statistically measured
- Software products are usually much less reliable than other engineering products

Robustness

- A software product is robust if it behaves reasonably even in unanticipated or exceptional situations
- A correct software product need not be robust
  - Correctness is accomplished by satisfying requirements
  - Robustness is accomplished by satisfying unstated requirements

Question on Correctness and Robustness

All correct programs are robust, but all robust programs are not necessarily correct. Is this statement True or False?

A. True

B. False

Performance

What are some ways you could measure software performance?

What are some ways you could specify performance requirements to make them unambiguous and verifiable?