CS 4530: Fundamentals of Software Engineering
Module 6.1: Software Development Processes

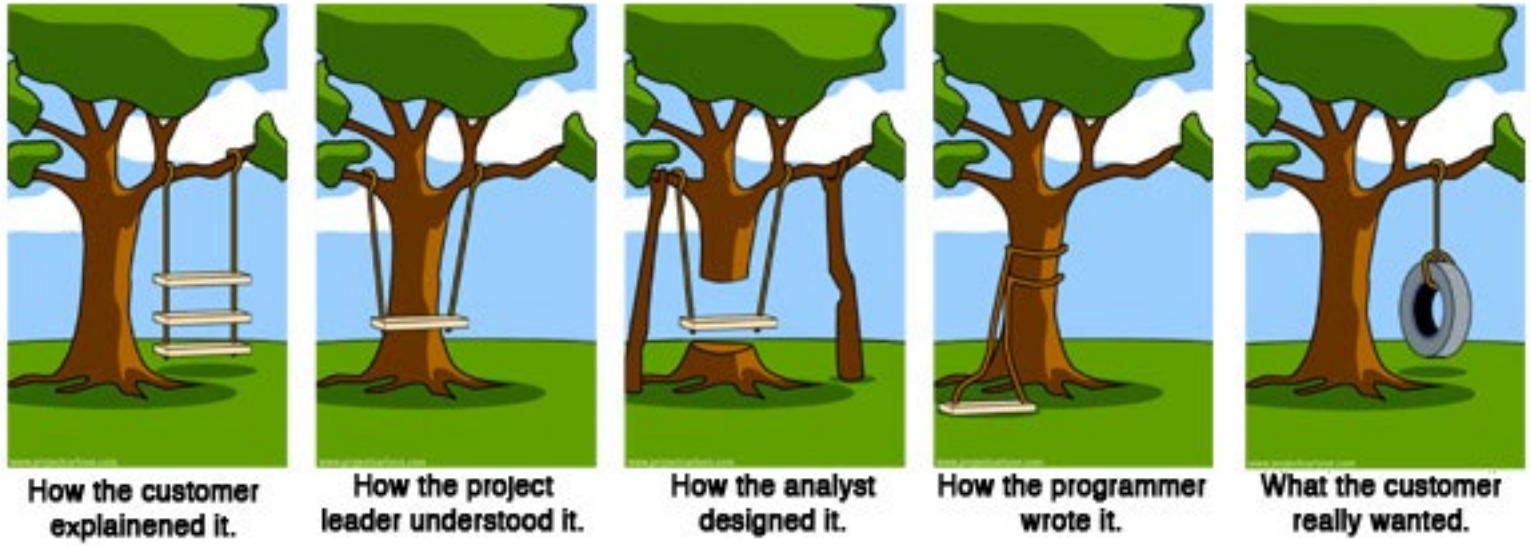
Adeel Bhutta, Jan Vitek and Mitch Wand

Khoury College of Computer Sciences

Learning Goals for this Lesson

- At the end of this lesson, you should be able to
  - Know the basic characteristics of the waterfall and agile software development models
  - Be able to explain when the each model is appropriate and when it is not
  - Understand how the waterfall and agile models manage risk
  - Be able to explain how agile process instill quality, including through test driven development

Review:
How to make sure we are building the right thing

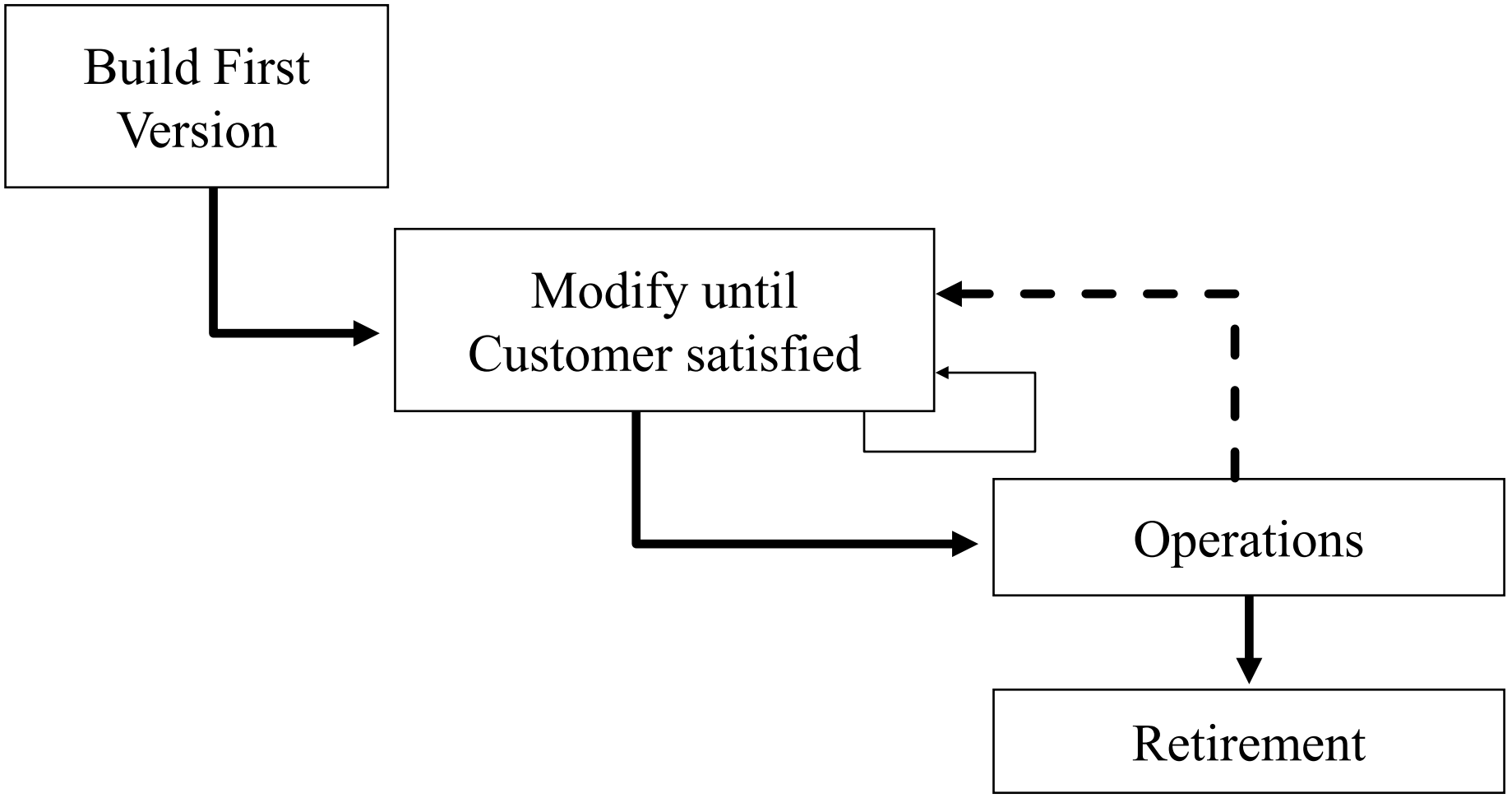


Requirements Analysis

Planning & Design

Implementation

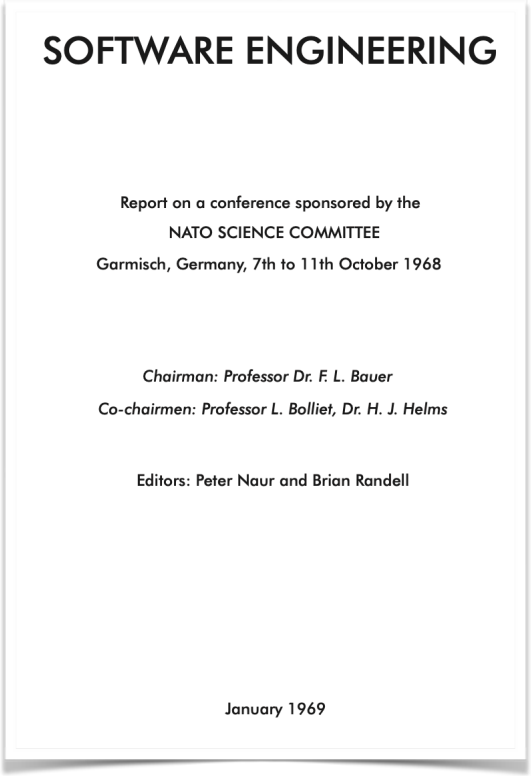
Software Process: Code + Fix



A brief history of software planning

NATO conference on Software Engineering + Outcomes

- Software was very inefficient
- Software was of low quality
- Software often did not meet requirements
- Projects were unmanageable and code difficult to maintain
- Software was never delivered


SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

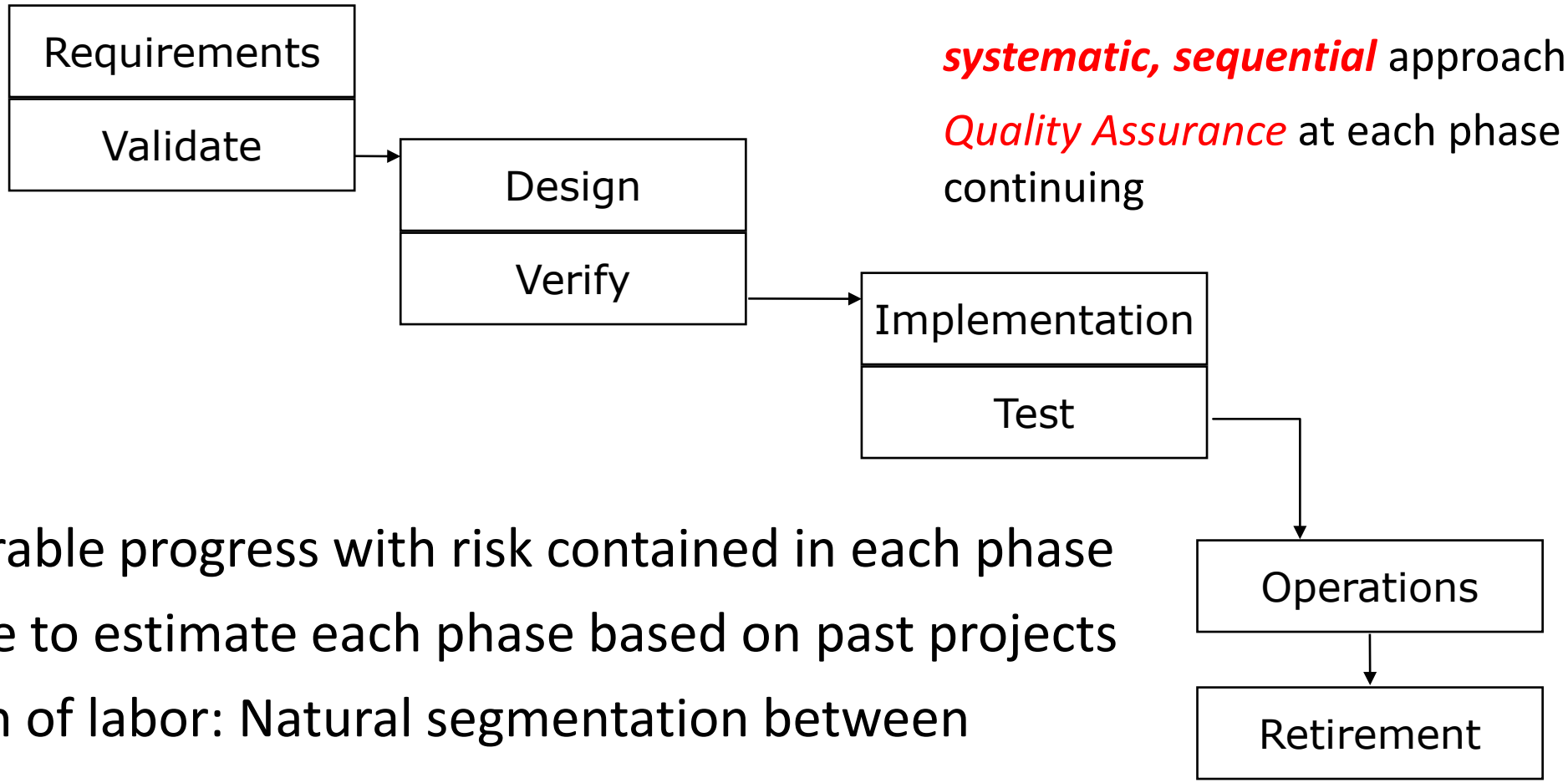
Chairman: Professor Dr. F. L. Bauer
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

January 1969

A call to action: We must study *how to build software*

The Waterfall Process Manages Risk by Phase-Boxing



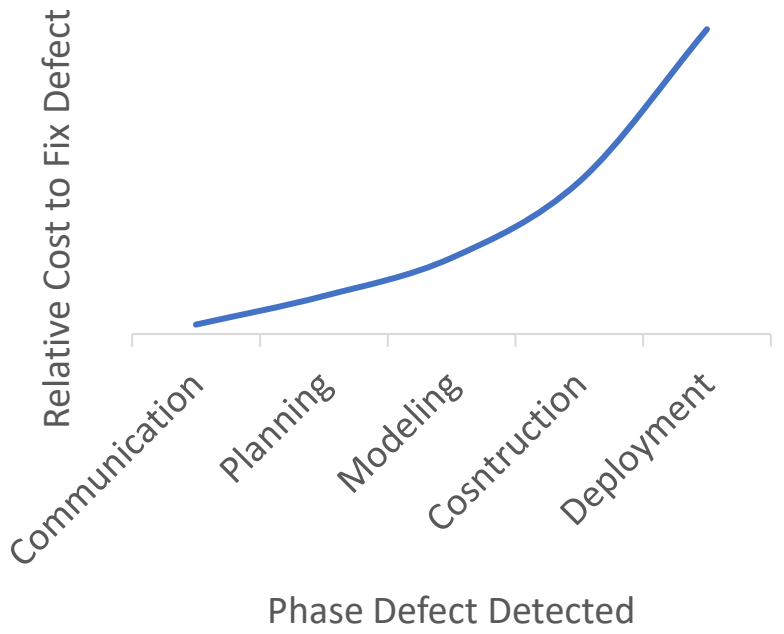
systematic, sequential approach

Quality Assurance at each phase before continuing

- Measurable progress with risk contained in each phase
- Possible to estimate each phase based on past projects
- Division of labor: Natural segmentation between phases

Phase-boxing manages risk by discovering defects early

- The graph shows cost to make a change or fix a defect grows exponentially over time
- By carefully engineering the communication, planning, and modeling phases, we can find hidden defects in those stages, rather than later on in the construction and deployment phases.



Waterfall Model: Applications

- What projects would this work well in?
  - Projects with tremendous uncertainty
  - Projects with long time-to-market
  - Projects that need extensive QA of requirements and design
  - Projects for which the expense of the planning is worth it
  - Classic examples: military/defense
    - Warship that needs to have component interfaces last 80 years
    - Spacecraft?

Waterfall Model adds process overhead

Since formal quality assurance happens at each phase, it's necessary to produce extremely detailed…

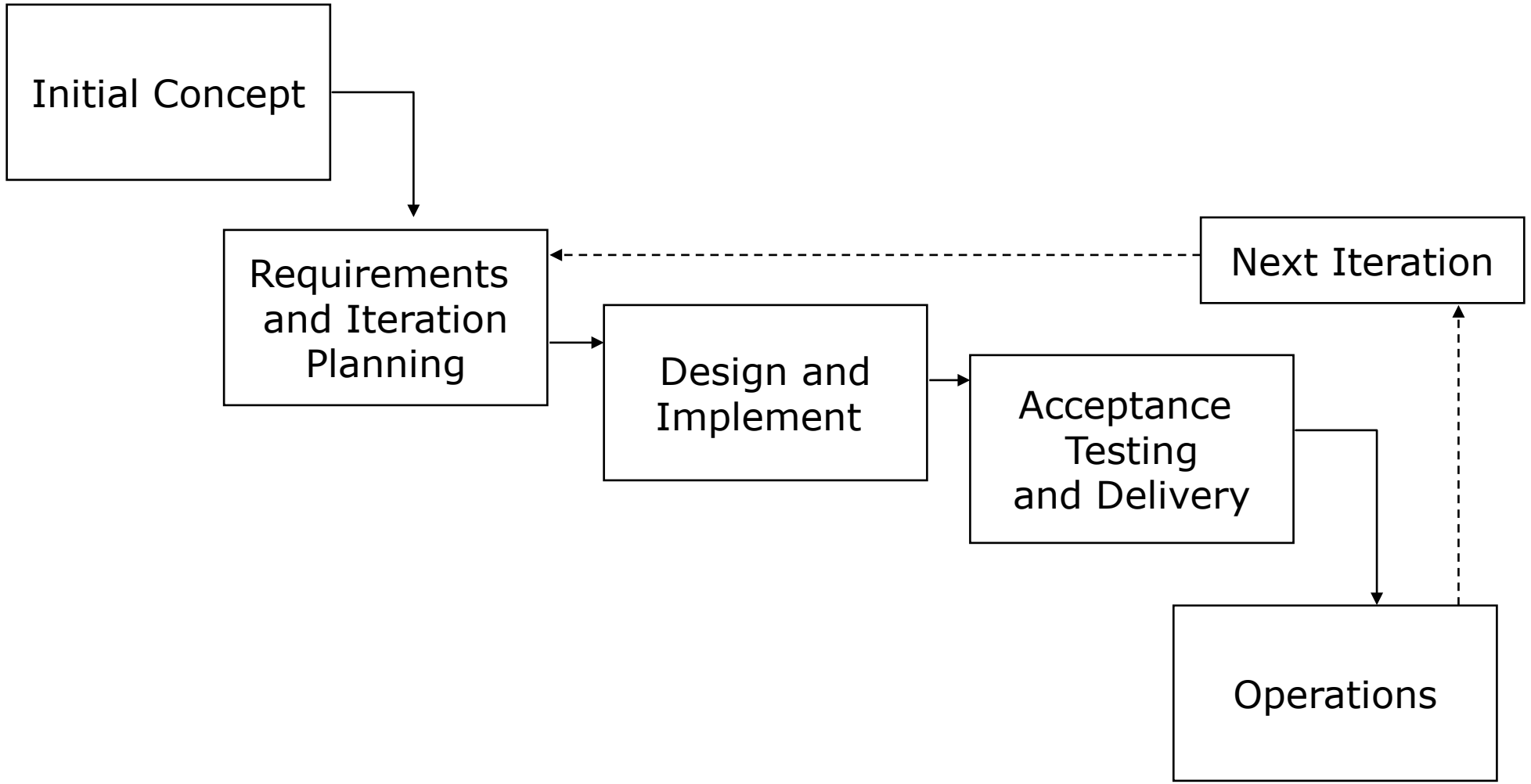
- Requirements documents
- Design documents
- Source code with documentation

Wasted productivity can occur through each phase's QA process

Waterfall Model Reduces Risk by Preventing Change

Traditional waterfall model: no way to go back “up”

Waterfall Variation: Iterative Process (~1980s)



The Agile Model Reduces Risk by Embracing Change (~2000)

- The Waterfall philosophy:
  - "The project is too large and complex, and it will take months (or years!) to plan, so once we come up with the plan, that plan can not change"
  - Reduce risk by proceeding in stages
- The Agile philosophy:
  - The project is too large and complex, it is unlikely that we will know exactly what we need right now, and to some extent, we are inventing something new. We think that as we make it, we will figure it out as we go”
  - Reduce risk by limiting time on any one stage; then reassess. (“time-boxing”). That way we won’t go too far in the wrong direction.

Agile Manifesto

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions	over processes and tools
Working software	over comprehensive documentation
Customer collaboration	over contract negotiation
Responding to change	over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

https://agilemanifesto.org

Agile Values Embrace Change

Compare to problems in waterfall:

- Requirements that become obsolete
  - Don’t make detailed requirements until you need them
- Elaborate architectural designs never used
  - Don’t design until you need
- Code that sits around not integrated and tested in production environment, eventually discarded
  - Integrate and test continuously
- Documentation produced per requirements, but never read
  - Don’t require documentation

Or only as much documentation as you really need.

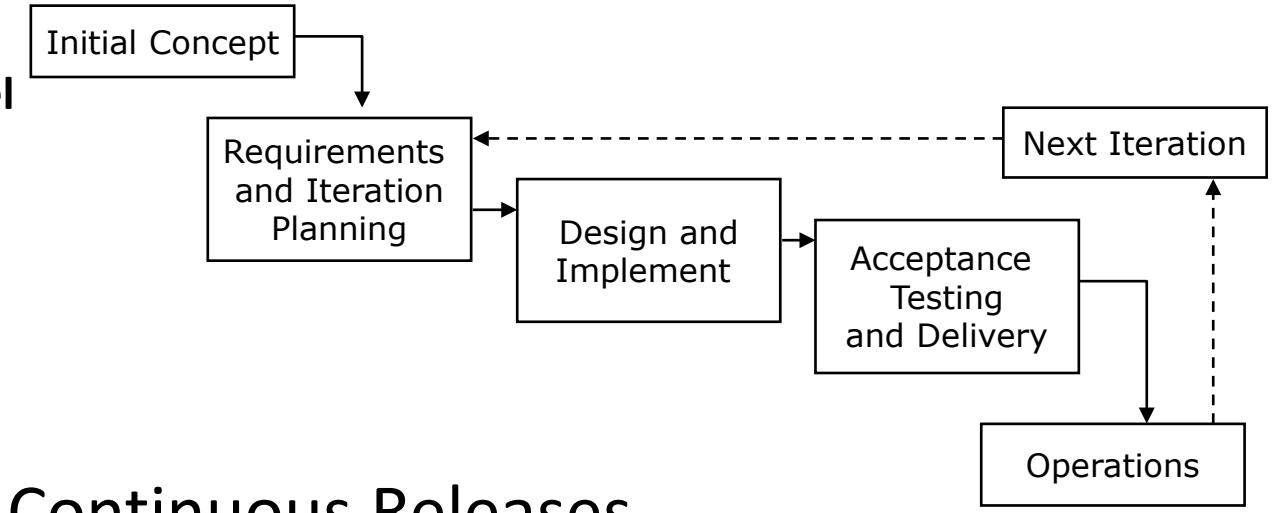
Agile Practice: Everyone is Responsible for Quality

- “Collective ownership”
- Requirements (user stories) are developed collaboratively with customer, and are *negotiable* (INVEST qualities)
- Functional and non-functional correctness is checked *on the cheap*, and often
- Developers improve code anywhere in the system if they see the opportunity
- Many parallels with “Toyota Process System;” a variety of other software processes developed in the 90’s share these basic values

Agile Processes are Iterative

Agile Process Model

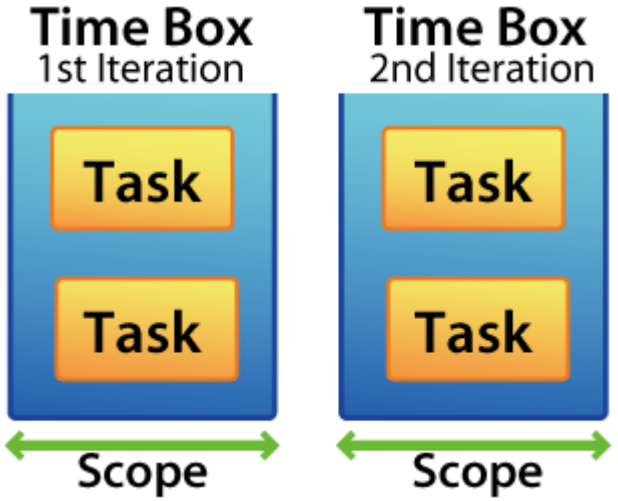
Iterative Waterfall Model



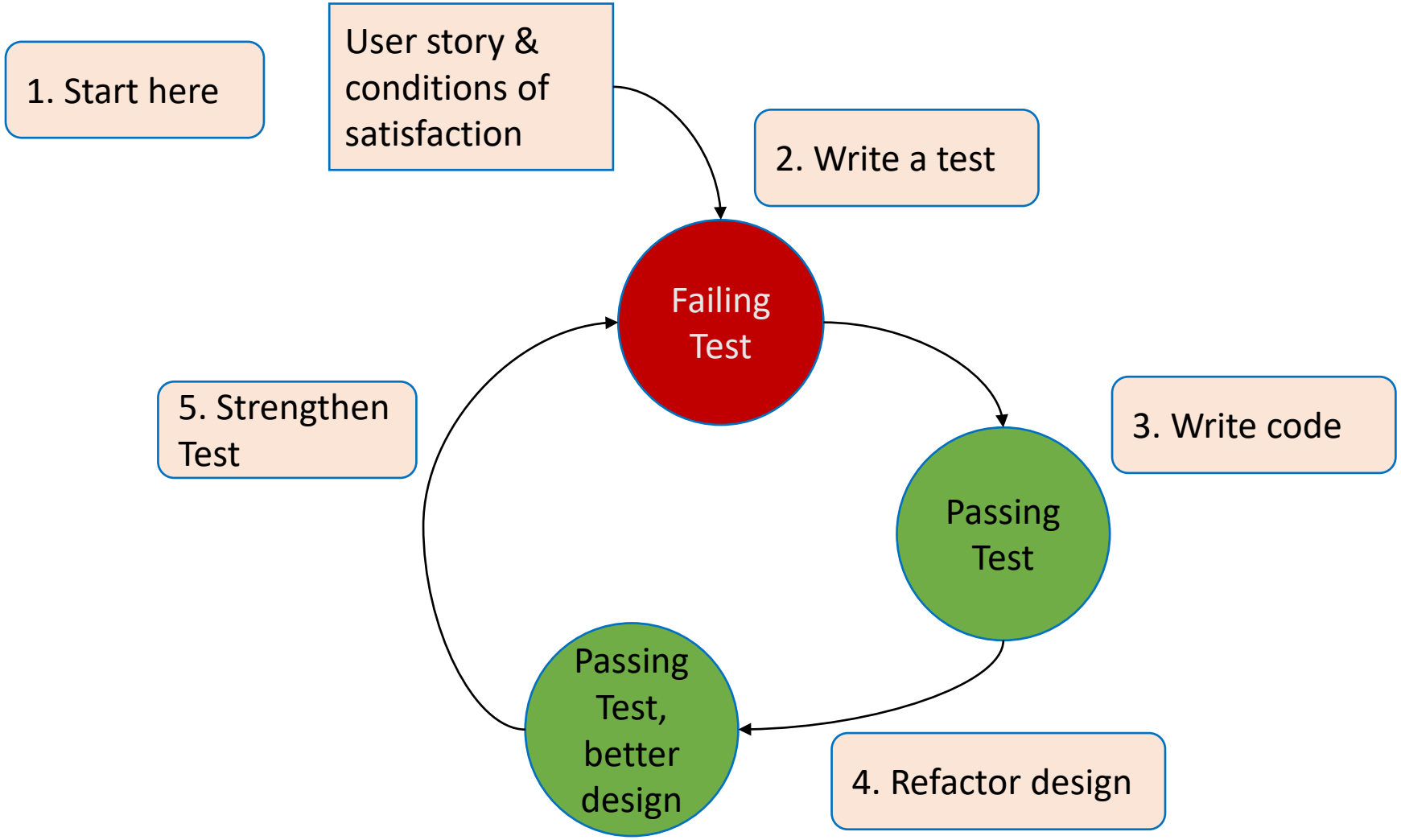
Key Idea: Small Continuous Releases

Agile Processes Reduce Risk by Time Boxing

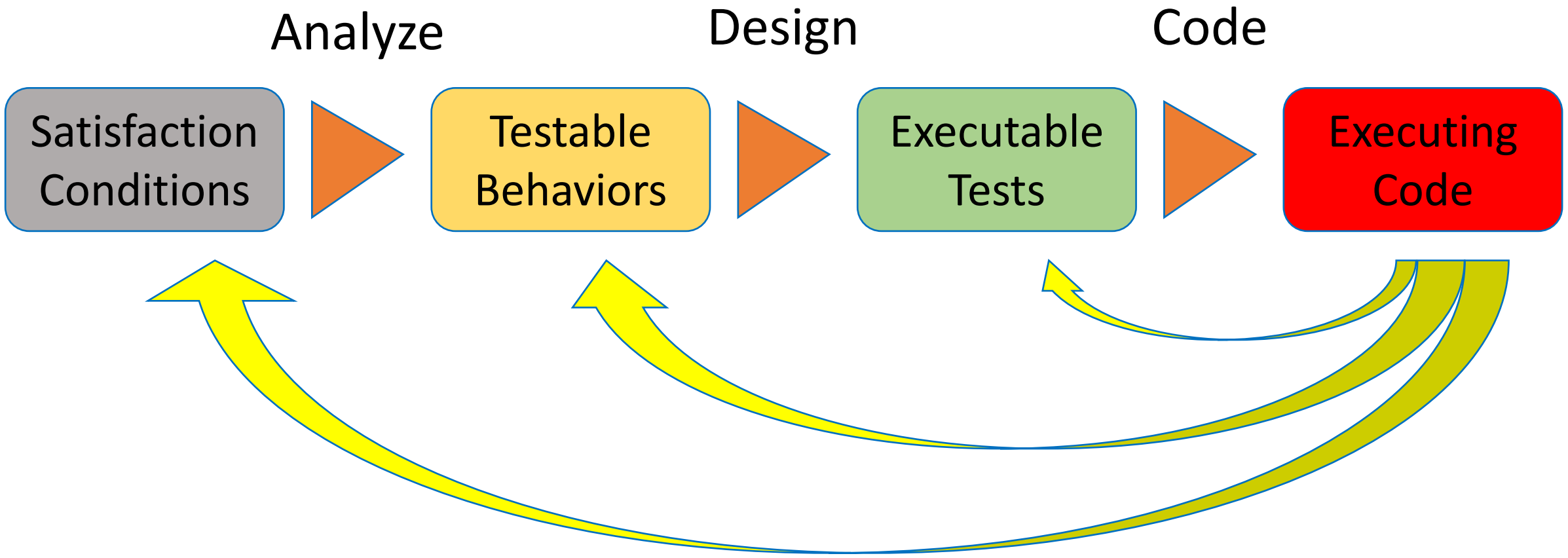
- Each “iteration” is called a “sprint”
- Each sprint has a fixed duration
- Scope of features in a sprint is determined by the team
- Key insight: planning might be a guess at first, but gets better with time
- More on agile planning & estimation in Module 6.2



Agile Practice: Test Driven Development (TDD)



The TDD Cycle (from Module 02)



Learning Goals for this Lesson

- You should now:
  - Know the basic characteristics of the waterfall and agile software development models
  - Be able to explain when the each model is appropriate and when it is not
  - Understand how the waterfall and agile models manage risk
  - Be able to explain how agile process instill quality, including through test driven development